Universitat Politècnica de Catalunya

Departament de Llenguatges i Sistemes Informàtics

Efficient Techniques in Global Line Radiosity

Memòria presentada per **Francesc Castro Villegas** a la Universitat Politècnica de Catalunya amb la finalitat d'obtenir el grau de **doctor en Informàtica**

Director: Mateu Sbert i Casasayas

Barcelona, octubre de 2002

Acknowledgments

First of all, thanks to Mateu Sbert, the director of this thesis, for his strong implication and dedication along these years.

I would also like to thank Xavier Pueyo, Miquel Feixas, László Neumann, László Szirmay-Kalos and Jaume Rigau for their collaboration along this thesis. Special thanks also to Joaquim Gelabertó for his valuable comments. Also thanks to the GGG (Graphics Group of Girona), the Department of IMA in the University of Girona, and to Núria Pla, my tutor in the UPC.

Finally I am very grateful to my family, and specially to my wife Isabel, that has always encouraged me in the bad moments, and to my daughter Sílvia, that has increased my motivation in the last period of this work.

The work that has led to this thesis has been funded in part by grants TIC 95-0614-C03-03, TIC 98-0586-C03-02, and TIC 2001-2416-C03-01 from the Spanish Government, and grant number 2001SGR-00296 from the Catalan Government.

Contents

Chapter 1

Introduction

One important topic in Computer Graphics is the computation of the global illumination of a scene to obtain realistic images [20]. We will focus in this thesis on *radiosity techniques* [14, 55]. Radiosity techniques, borrowed from thermal engineering in the eighties, determine the value of radiant power between the surfaces in a scene. They deal basically with diffuse environments, in which the illumination is independent of the point of view.

Monte Carlo (or stochastic) methods are frequently used in radiosity. They have an important drawback: their high computational cost. The reduction of this cost is the objective of many lines of research and also of this thesis. We will deal in this thesis with Monte Carlo approaches to radiosity based on random global lines, i.e. lines that are cast at the level of the whole scene. This differs from local lines, which are cast from a specific surface (see Fig. 1.1).

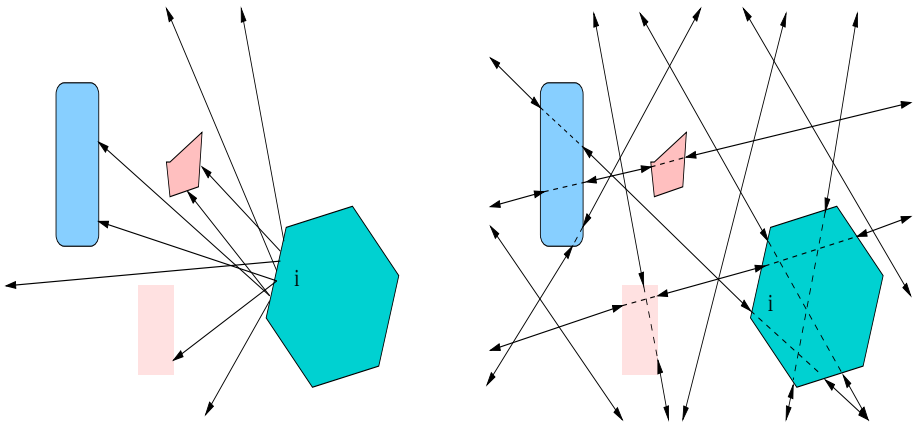


Figure 1.1: *(left) Local (to surface i) lines. (right) Global lines.*

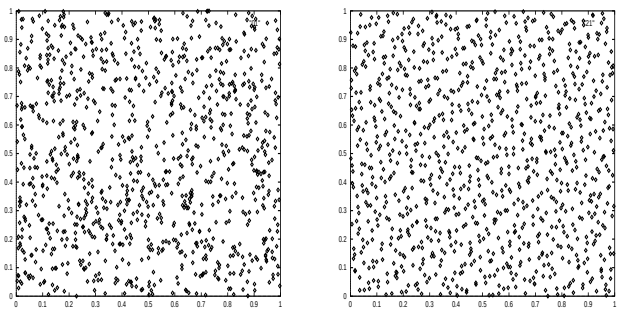
Efficient techniques for these methods are presented here. These techniques either reduce the computational cost of casting the global lines, or reduce the number of these lines needed to obtain an acceptable accuracy, or improve the quality of the final image with negligible increase of cost. Next we briefly describe the contents of each of the chapters in which this thesis is organized.

1.1 Chapter 2: Previous Work

This chapter presents, on the one hand, the basis of the global illumination and the radiosity method, and, on the other hand, the techniques of Monte Carlo and quasi-Monte Carlo integration. Furthermore it explores different applications of Monte Carlo methods in the context of radiosity, specially reviewing global line radiosity algorithms like Multipath. Finally it presents the previous applications of quasi-Monte Carlo integration to realistic rendering and specially to radiosity.

1.2 Chapter 3: Use of quasi-Monte Carlo sequences in the Multipath context

Quasi-Monte Carlo integration, based on the use of low discrepancy sequences, that is, sequences of values specially designed to be more evenly distributed on the domain (see Fig. 1.2) is incorporated here to the context of the Multipath algorithm [51]. We present in this chapter our research [6, 9, 10] on the the use of different quasi-Monte Carlo sequences in the generation of the samples needed by the Multipath algorithm. Reductions of the computational cost to nearly the half have been obtained.



(a) (b)

Figure 1.2: *(a) Monte Carlo random points (b) Quasi-Monte Carlo low discrepancy points*

1.3 Chapter 4: Hierarchical improvement to a global Monte Carlo method for form factors computation

This chapter presents a hierarchical strategy to improve the performance of the global Monte Carlo algorithm [47] in which form factors are estimated. This strategy is based on the use of a hierarchy of sub-scenes in which the main scene is organized [8]. This allows us to use densities of global lines adapted to each sub-scene, driving to noticeable reductions on the error of the estimated form factors. Furthermore, quasi-Monte Carlo sequences are incorporated to this algorithm, resulting in an additional gain.

1.4 Chapter 5: Hierarchical Transmittance-based Multipath

In this chapter we introduce a hierarchy of sub-scenes to the Multipath algorithm [11]. The idea is similar to the one in chapter 4, that is, to cast more random lines where they are more needed, but it also incorporates transmittances, that suppose an additional information for each sub-scene obtained in a preprocess and used later by the algorithm. Reductions of the cost by one half have been obtained in our tests.

1.5 Chapter 6: Extended ambient term

This chapter presents a new technique [7] based on the idea of ambient radiosity, but adding some geometric considerations, like the orientation of each surface. This technique is applicable to the radiosity context, but it is also possible to use it in the global illumination context. Some nice effects in the final image, like color bleeding or color shifting, have been obtained with this strategy. The use of a hierarchy of sub-scenes like the one presented in chapter 5 gives an additional gain to the results obtained with the extended ambient term.

1.6 Chapter 7: Conclusions and future work

In this chapter we summarize the conclusions of all the work presented in this thesis, and present the main lines of future research.

Chapter 2

Previous Work

2.1 Radiosity and global illumination

This thesis deals with *global illumination* in computer graphics. Global illumination models consider not only the direct light coming from the light sources but also indirect lighting. These models are the most frequently used to obtain realistic images, because they consider several levels of light reflection. Global illumination approach is much more accurate than local illumination, in which only direct lighting is considered.

The problem of global illumination is solved in computer graphics by simulating the inter-reflection of light between all the surfaces in a scene [55]. One of the most common approaches to global illumination and the one this thesis deals with is the *radiosity method*. Although some non-diffuse approaches have been presented [26, 54], the radiosity method basically deals with diffuse reflectors. Diffuse reflectors (also called Lambertian) reflect the same intensity of radiance in all outgoing directions. This means that the BRDF (Bidirectional Reflection Distribution Function) is independent of the direction, namely it is a constant (see Figure 2.1). This implies that the obtained solutions are view independent, which is specially interesting in walk-through navigation systems, virtual reality, etc.

2.1.1 The rendering equation

The rendering equation (2.1) [27] expresses the light transport in a closed environment

$$L(x,\omega) = L_e(x,\omega) + \int_{\mathcal{S}} \rho(x,\omega,-\omega')L(x',\omega')G(x,x')dA' \tag{2.1}$$

where (see Fig. 2.2)

- x, x' are points on surfaces of the environment
- ω, ω' are outgoing directions at x, x'
- $L(x,\omega), L(x',\omega')$ are the total (reflected+emitted) exiting radiance at points x and x' in directions ω and ω' respectively.

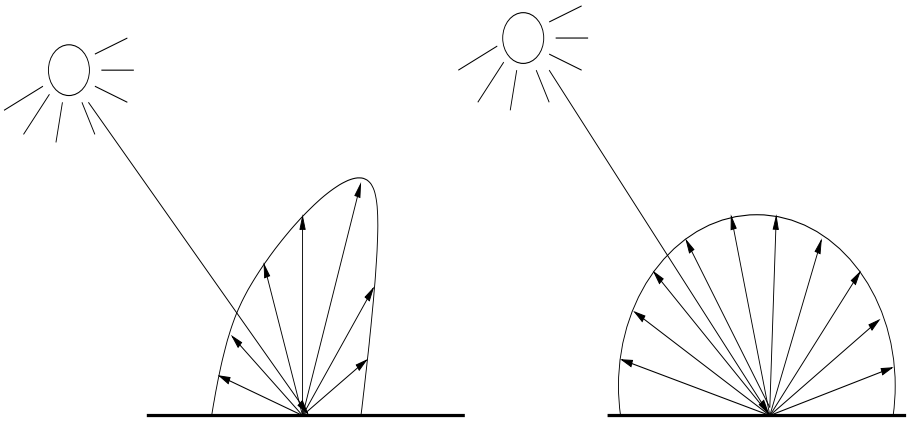


Figure 2.1: *Bidirectional Reflection Distribution Function (BRDF). (left) General reflectance. (right) Diffuse reflectance.*

- $L_e(x, \omega)$ is the emitted radiance at point x in direction ω
- $\rho(x, \omega, -\omega')$ is the bidirectional reflectance distribution function (BRDF). It is the fraction of the incident radiance in direction ω' reflected in direction ω at point x
- $G(x, x')$ is the *geometric term*, described below (equation 2.2)
- dA' is an area differential at point x'
- $\mathcal{S}$ is the set of surfaces that form the environment

Note that the rendering equation expresses the exiting radiance of each point in the environment. The radiance is the angular flux density of energy, that is, the power per unit area and per unit solid angle ($\frac{W}{m^2 sr}$). This equation can be interpreted as follows: the exiting radiance of a point in a direction is equal to the emitted radiance plus the reflected radiance, and the reflected radiance is the sum of all the contributions from all the points in the environment.

The geometric term $G(x, x')$ is the crucial part in the rendering equation, because it involves the visibility queries:

$$G(x, x') = \frac{V(x, x') cos\theta cos\theta'}{r^2} \tag{2.2}$$

where θ and θ' are the angles between the directions ω, ω' and the normals to the surface at x, x', respectively, r is the distance between x and x' and $V(x, x')$ is the visibility function, equal to 1 if x and x' are mutually visible and 0 otherwise.

Fig. 2.2 represents the different elements of the rendering equation.

2.1.2 The radiosity system of equations

Considering diffuse (or Lambertian) environments, the BRDF depends only on the reflection point x, and it does not depend on direction. This allows to derive

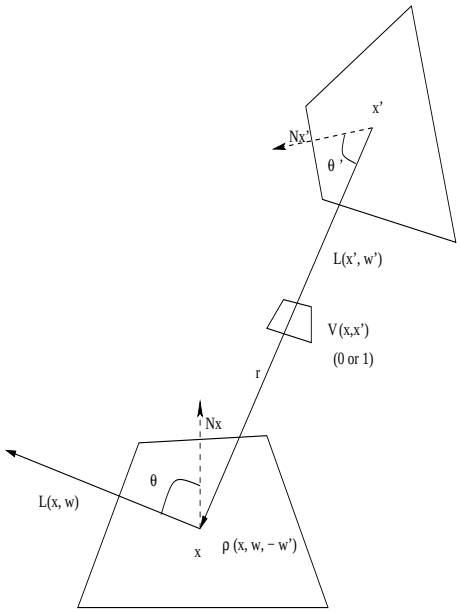


Figure 2.2: *The rendering equation geometry.*

the radiosity equation (2.3) from the rendering equation. Note that we consider radiosities (exiting power per unit area ($\frac{W}{m^2}$)) instead of radiances. The radiosity equation expresses the radiosity $B(x)$ of every point in the scene [14, 55]:

$$B(x) = E(x) + \rho(x) \int_{\mathcal{S}} B(x') \frac{G(x, x')}{\pi} dA' \tag{2.3}$$

where

- $B(x), B(x')$ are the radiosity of points x and x' respectively.
- $E(x)$ is the emittance of point x, that is, the primary power per unit area (only non-zero for light sources)
- $\rho(x)$ is the reflectance of point x (value between 0 and 1 that expresses the fraction of incident power that is reflected)
- $G(x, x')$ is the geometric term
- dA' is an area differential at point x'

Note that in this equation the radiosity of each point is expressed as the sum of the emittance $E(x)$ (primary power per unit area) and a term that represents the incoming radiosity (from the rest of points in the scene) times the reflectance of x (fraction of power that is reflected by point x). Equation (2.3) has a closed solution only for very simple environments. In general we have to use numerical methods to solve it. Some of the earlier radiosity approaches can be seen in [22, 13, 12, 66].

A first step to solve equation (2.3) is to consider finite elements: all the surfaces in the scene are subdivided into patches. A particular and most commonly used case considers the radiosity, reflectance and emittance to be constant along each patch. This drives us to equation (2.4), the discrete version of the rendering equation [55].

$$B_i = E_i + \rho_i \sum_{j=1}^{N} F_{ij} B_j \tag{2.4}$$

where

- B_i is the (outgoing) radiosity of patch i
- E_i is the emittance of patch i
- ρ_i is the reflectance of patch i
- N is the number of patches
- F_{ij} is the form factor from patch i to patch j, that will be described in next section.

The radiosity method consists of solving the system of linear equations (2.4). Observe that coefficients F_{ij} are in general unknown. The critical point in this system is thus the computation of form factors that involves the visibility queries.

2.1.3 The Form Factors

The form factor F_{ij} from patch i to patch j is the fraction of energy that leaving patch i goes directly to patch j. The form factor F_{ij} can be expressed [55] as a quadratic integral:

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} V(x_i, x_j) dA_j dA_i \tag{2.5}$$

where A_i and A_j are, respectively, the areas of patches i and j, θ_i and θ_j are the angles between the line that joins dA_i and dA_j at points x_i, x_j respectively, and the respective normal vectors, and $V(x_i, x_j)$ is the binary visibility function between dA_i and dA_j (see figure 2.3). Note that form factors only depend on the geometry of the scene. We can also consider the form factor integrating on the hemisphere instead of over patch j. Since differential of solid angle is $d\omega = \frac{cos\theta_j}{r^2} dA_j$, we have the patch to hemisphere form

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{\Omega} \frac{\cos\theta_i}{\pi} V_j(x_i, \omega) d\omega dA_i \tag{2.6}$$

where $V_j(x_i, \omega)$ is the binary visibility function that indicates if patch j is visible from x_i in direction ω. This form is useful to compute at once all the form factors from patch i. Since $d\omega$ can be expressed in polar coordinates as $sin\theta d\theta d\psi$, this integral can also be expressed as

$$F_{ij} = \frac{1}{\pi A_i} \int_{A_i} \int_{\theta} \int_{\psi} V_j(x_i, \theta, \psi) \cos\theta \sin\theta d\theta d\psi dA_i \tag{2.7}$$

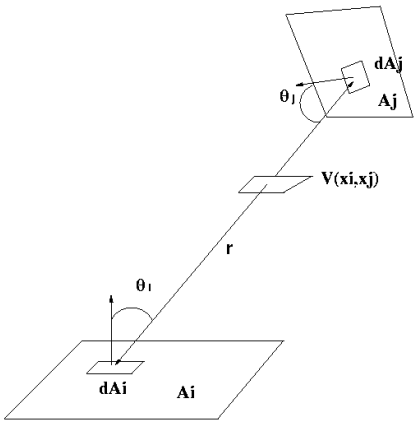


Figure 2.3: *Form factor geometry.*

where $V_j(x_i, \theta, \psi)$ is equal to 1 if patch j is visible from dA_i in direction (θ, ψ).

Two important properties of form factors can be easily derived. The first one (2.8) is a reciprocity relation. The second one (2.9) can be stated as the conservation of flux of energy:

$$A_i F_{ij} = A_j F_{ji} \quad \forall (i,j) \tag{2.8}$$

$$\sum_{j=1}^{N} F_{ij} = 1 \quad \forall i \tag{2.9}$$

The reciprocity relation allows a new formulation of the radiosity equation (2.4) considering power instead of radiosity. It is the power equation (2.10), that makes more evident the physical meaning of form factors, because it multiplies P_j, the outgoing power from patch j, by form factor F_{ji}, this is, the fraction of the power leaving patch j that reaches patch i:

$$P_i = \phi_i + \rho_i \sum_j F_{ji} P_j \tag{2.10}$$

where

- P_i is the outgoing power of patch i, $P_i = B_i A_i$
- ϕ_i is the emitter power of patch i (only non-zero if i is a source), $\phi_i = E_i A_i$

There is no closed form solution for form factors (except for very simple shapes without occlusions), thus deterministic numerical solutions have been developed. For instance, [13] presents a coarse approximation, the hemicube method, in which patch i is covered by an hemicube subdivided into pixels. Patch j is then projected over this hemicube. Other numerical methods are referenced in [14, 55].

Methods that compute and store the form factors and later solve the equation system are referred to as *full matrix* methods, but in most radiosity algorithms

either it is enough to compute one row at a time (progressive methods [66]) or it is not necessary to explicitely compute the form factors (Monte Carlo methods [35, 51, 63]), in this way avoiding the $O(n^2)$ storage requirements.

2.2 Monte Carlo and Quasi-Monte Carlo methods

2.2.1 Basic concepts

Random variable, probability density function (pdf) and distribution function

A random variable X is a variable whose value is not deterministic but stochastic. We distinguish between discrete and continuous random variables. We are here interested in continuous random variables, that can take an infinite uncountable number of values. For a continuous random variable, there exists a continuous and positive defined function $f(x)$ that describes the probability of variable X to take values. More formally, we have

$$Prob(a \leq X \leq b) = \int_a^b f(x)dx \tag{2.11}$$

being equal to 1 the integral between $-\infty$ and $+\infty$. Such a function $f(x)$ is referred to as the *probability density function* (pdf) of random variable X. We also define the *distribution function* $F(x)$ of X as

$$F(x) = Prob(X \leq x) = \int_{-\infty}^{x} f(u)du \tag{2.12}$$

Note that the derived function of the distribution function $F(x)$ is the pdf $f(x)$. The idea of pdf and distribution function can be extended to higher dimensional integration domains. A pdf most used in Monte Carlo is the constant one. A random variable has *uniform distribution* if its pdf is constant on its domain.

Expected value of a random variable

Given a continuous random variable X, with pdf $f(x)$, we define its *expected (or mean) value* in the following way

$$E(X) = \int_{-\infty}^{+\infty} xf(x)dx \tag{2.13}$$

The concept of expected value can easily be extended to higher dimensions. Next we enumerate some properties of the expected value, X and Y being random variables

$$E(cX) = cE(X) \qquad \forall c \in R \tag{2.14}$$

$$E(X+Y) = E(X) + E(Y) \tag{2.15}$$

$$E(XY) = E(X)E(Y) \quad if\ X\ and\ Y\ indep. \tag{2.16}$$

$$E(g(X)) = \int_{-\infty}^{+\infty} g(x)f(x)dx \tag{2.17}$$

From the first two properties (2.14,2.15) we have that the expected value is a linear operator. Third property (2.16) is valid if X and Y are independent. Last property (2.17) establishes the expected value for a random variable $g(X)$ defined in function of X.

Variance of a random variable

Given a continuous random variable X, we define its *variance* as

$$V(X) = E((X - E(X))^2) \tag{2.18}$$

This means that the variance is the expected value of the quadratic error. The variance is a measure of dispersion of the random variable. Next we enumerate some properties of the variance, X and Y being random variables

$$V(X) = E(X^2) - E^2(X) \tag{2.19}$$

$$V(X+Y) = V(X-Y) = V(X) + V(Y)\ if\ X,Y\ indep. \tag{2.20}$$

$$V(cX) = c^2V(X) \quad \forall c \in R \tag{2.21}$$

Estimating the expected value of a random variable

In practice, we have random variables from which we do not know their expected value. It is possible to estimate this expected value taking samples from the random variable and calculating their arithmetic mean. The mean of a set of samples from X is an *estimator* of its expected value $E(X)$. Moreover, if we consider the random variable resulting of taking the means of different sets of samples from X, its expected value is equal to $E(X)$. In other words, the mean is an *unbiased estimator* of $E(X)$.

2.2.2 Random sequences of $[0, 1)$ values

Before starting to deal with Monte Carlo and quasi-Monte Carlo methods, it is necessary to briefly discuss about what can be considered a random sequence of $[0, 1)$ values. Note that all the values used in computer simulations are generated by computer programs, so they are in fact pseudo-random rather than random. We can ask if they are valid for our simulations. Let us review some definitions.

First, let us consider an infinite sequence $< U_i >$ of $[0, 1)$ values. We define the sequence $< U_i >$ to be *equidistributed* (or *uniformly distributed*) if, for any values a and b such that $0 \leq a < b \leq 1$ it satisfies [33]

$$\lim_{n\to\infty} \frac{\nu(n)}{n} = b - a \tag{2.22}$$

where $\nu(n)$ is the number of values from $U_0, ...U_{n-1}$ that belong to the interval $[a, b)$. Note that here we base the intuitive idea of probability on the frequency of occurrence. The following definition [33] generalizes this idea, introducing the concept of probability for an infinite sequence to satisfy a property. Let $S(n)$ be a statement about the integer n and the sequence $< U_i >$. We define the probability of $S(n)$ to be true in the following way:

$$Prob(S(n)) = \lambda \text{ if } \lim_{n\to\infty} \frac{\nu(n)}{n} = \lambda \tag{2.23}$$

where $\nu(n)$ states for the number of values from $U_0, ...U_{n-1}$ that satisfy property S.

1-uniform sequences

In terms of the previous definitions, and a suitably defining property $S(n)$, we can define a sequence $< U_i >$ to be equidistributed on I if it satisfies, for all real numbers a, b with $0 \leq a < b \leq 1$, the following

$$Prob(a \leq U_i < b) = b - a \tag{2.24}$$

for any value U_i of the sequence and for any interval $[a, b)$ contained in $[0, 1)$. This kind of sequences are also referred to as 1-uniform or 1-distributed. This property is a necessary requirement for an infinite sequence to be considered as random. But this is not enough. Let us consider, for instance, the infinite sequence composed by $(\frac{1}{2}U_0, \frac{1}{2}+\frac{1}{2}V_0, \frac{1}{2}U_1, \frac{1}{2}+\frac{1}{2}V_1, ...)$, $< U_i >$ and $< V_i >$ being 1-uniform sequences [33]. This sequence satisfies the property (2.24), but since the values in even positions are lower than 0.5 and the ones in odd positions are greater or equal than 0.5, it cannot be considered a random sequence (*Note: the concept of random sequence is not trivial, and it involves philosophical questions. For a detailed discussion on randomness, see [33].*)

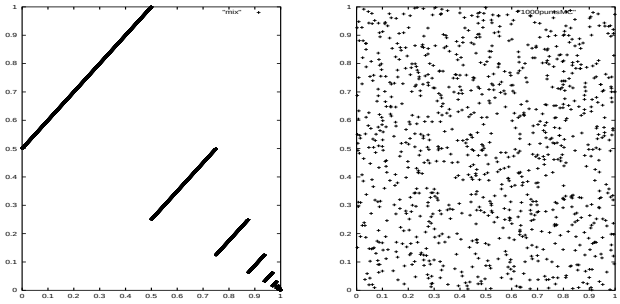
2-uniform sequences

An infinite sequence $< U_i >$ of values in I is said to be *2-uniform* if it satisfies

$$Prob(a_1 \leq U_i < b_1, a_2 \leq U_{i+1} < b_2) = Prob(a_1 \leq U_i < b_1)Prob(a_2 \leq U_{i+1} < b_2) =$$

$$= (b_1 - a_1)(b_2 - a_2) \tag{2.25}$$

for any pair of consecutive values U_i and U_{i+1} of the sequence. Fig. 2.4 shows, on the left, a plot of 2D points obtained from an 1-uniform but not 2-uniform sequence, and, on the right, a plot of 2D points from a 2-uniform sequence.



(a) (b)

Figure 2.4: *2D points from a (a) 1-uniform sequence (b) 2-uniform sequence.*

Note that the idea of 2-uniformity is stronger than the idea of 1-uniformity. It implies not only 1-uniformity but also independence between each pair of consecutive values. Note for instance that the previously seen infinite sequence $(\frac{1}{2}U_0, \frac{1}{2} + \frac{1}{2}V_0, \frac{1}{2}U_1, \frac{1}{2} + \frac{1}{2}V_1, ...)$, with $< U_i >$ and $< V_i >$ being 1-uniform sequences, is not 2-uniform.

k-uniform and ∞-uniform sequences

We can extend the concept of 2-uniformity to k-uniformity for any $k > 0$. An infinite sequence $< U_i >$ of values in I is said to be *k-uniform* if it satisfies

$$Prob(a_1 < U_i < b_1, .., a_k < U_{i+k-1} < b_k) =$$

$$= Prob(a_1 < U_i < b_1)...Prob(a_k < U_{i+k-1} < b_k) =$$

$$= (b_1 - a_1)...(b_k - a_k) \tag{2.26}$$

From here we note that k-uniformity of a sequence implies l-uniformity for each l less than k.

There also exist ∞-uniform sequences [33], that is, sequences of values in I that are k-uniform for any $k > 0$. These sequences are also known as *completely uniformly distributed* [40, 32]. This is one of the possible definitions of a random sequence [33]. In a ∞-uniform sequence, the subsequent values are independent between them: each value is independent from the previous ones. Sequences of values from computer standard random generators appear to behave in this way [59, 32]: they are ∞-uniform with probability 1.

2.2.3 Sequences of d-dimensional points uniformly distributed on I^d

We extend the idea of equidistribution (see section 2.2.2) to any dimension d. A sequence of d-dimensional points $< p_i >$ in I^d is said to be *uniformly distributed* (also equidistributed) on I^d if, for any box

$$A = [a_1, b_1] \times ... \times [a_d, b_d] \qquad in\ I^d \tag{2.27}$$

it satisfies the following identity

$$\lim_{n\to\infty} \frac{|\{p_1, ..., p_n\} \cap A|}{n} = |A| \tag{2.28}$$

where $|A|$ is the volume of A. In other words, this means that the probability of a point in the sequence to belong to any box $A \in I^d$ is equal to the volume of the box. Note that, in dimension 1, this definition is equivalent to the one of 1-uniformity seen in 2.2.2.

Note that it is possible to obtain an uniformly distributed sequence of d-dimensional points from a d-uniform sequence $< U_i >$ of values in I in the following way [33]:

$$p_i = (U_i, U_{i+1}, .., U_{i+d-1}) \tag{2.29}$$

For instance, if we consider dimension 3 and have a 3-uniform sequence of values in $I < U_i >=< U_1, U_2... >$, we can obtain the following sequence of 3d points
$p_1 = (U_1, U_2, U_3)$
$p_2 = (U_2, U_3, U_4)$
$p_3 = (U_3, U_4, U_5)$
....

that will be uniformly distributed on I^3.

2.2.4 Monte Carlo methods

The Monte Carlo methods [24] are stochastic methods that solve mathematical problems by means of the simulation of random variables. Basically, the idea is to obtain a sequence of independent random samples from an uniform random variable and to consider the mean of the results. The Monte Carlo method is used to estimate integrals for which no analytic solution can be found. This is referred to as *Monte Carlo integration.*

More accurately, the Monte Carlo method allows to integrate a function $g(x)$ on a domain D by generating a sequence of independent samples on D according to a probability density function (pdf) $f(x)$. The value of the integral can be seen as the expected value of the random variable $\frac{g(x)}{f(x)}$ with pdf $f(x)$ (2.30), and this can be estimated by sampling the variable on D using $f(x)$ as pdf, obtaining the unbiased estimator (2.31):

$$I = \int_D g(x)dx = \int_D \frac{g(x)}{f(x)} f(x)dx = E_{f(x)}\left[\frac{g(x)}{f(x)}\right] \tag{2.30}$$

$$I \approx I_N = \frac{1}{N} \sum_{k=1}^{N} \frac{g(x_k)}{f(x_k)} \tag{2.31}$$

The samples on D according to the density function $f(x)$ are usually obtained from the inverse of the distribution function $F(x)$ (2.12). This procedure is known as *inversion method* [40], and consists of computing the sequence of samples x_k from $F^{-1}(\xi_k)$. $< \xi_k >$ is a sequence of realizations of independent random variables with uniform distribution in I^d, d being the dimension of the integration domain. In practice, such a sequence $< \xi_k >$ can be obtained from the $[0, 1)$ values provided by the computer random generator. From here on, we will refer to these sequences of points as *Monte Carlo sequences*.

Note that expression (2.31) also converges for a broader kind of sequences, the uniformly distributed ones. This is, the only requirement for a sequence to guarantee the convergence of (2.31) is to be uniformly distributed on I^d [25] (see section 2.2.3). From this point of view note that the Monte Carlo sequences are considered ∞-uniform [59, 32], and thus they are valid to generate sequences of uniformly distributed points on any dimension. The use of expression (2.31) in the context of deterministic uniformly distributed sequences is known as quasi-Monte Carlo method and will be described in section 2.2.8.

2.2.5 Error in Monte Carlo integration

Monte Carlo methods are probabilistic, based on sampling values from random variables. The value of the integral is seen as an expected value, and variance must be considered. Quasi-Monte Carlo methods, (see section 2.2.8), although using the same expression (2.31), are totally deterministic numerical methods. Thus it has no sense considering neither expected values nor hence variance in their context.

Let us consider we are integrating a square integrable function, that is, a function that belongs to L^2 [40]. Then the error in the Monte Carlo estimation (or convergence rate) is proportional to $N^{-\frac{1}{2}}$, where N is the number of samples taken. As an example, this means that the number of samples has to be multiplied by 100 to reduce the error by one order of magnitude. The variance for the estimator (2.31), that is, the expected value of the quadratic error, is given by

$$V(I_N) = \frac{1}{N}\left(\int_D \frac{g(x)^2}{f(x)} dx - I^2\right) \tag{2.32}$$

2.2.6 Monte Carlo methods and the curse of dimensionality

Let us consider the problem of numerical integration in dimension d. If we use classical integration rules, as the Simpson's rule or the trapezoidal rule, a d-dimensional integral is considered as an iteration of one-dimensional integrals, so that there is a dependence on the dimension of the domain. The error bound is established as $O(N^{-1/d})$ [60]. This means that increasing the dimension d, the required number of samples to get a certain accuracy increases exponentially. This phenomenon is often called the *curse of dimensionality* or the *dimensional*

explosion [40, 64], and it represents a great limitation for classic integration methods when dealing with higher dimensions.

Note that Monte Carlo methods overcome the curse of dimensionality, because their performance does not depend on the dimension of the integral to solve. The error of Monte Carlo methods is established as $O(N^{-\frac{1}{2}})$, therefore it is independent of the dimension (the variance (2.32) is the expected value of the quadratic error, and it is $O(\frac{1}{N})$). From this fact Monte Carlo methods are a very general and powerful tool to solve integrals regardless of their dimension.

2.2.7 Importance sampling in Monte Carlo

We can see in equation (2.32) that the variance depends on the probability density function $f(x)$ used in the Monte Carlo sampling. It can be shown that the minimum variance is obtained taking $f(x) = \frac{|g(x)|}{I}$ [28]. Since the value of the integral is unknown, density functions that mimic the integrand have to be used. These functions are called *importance functions*. The sampling according to these importance functions is called *importance sampling*. In other words, importance sampling consists of sampling more points in the regions where $|g(x)|$ is greater. This technique is widely used in Monte Carlo methods.

2.2.8 Quasi-Monte Carlo methods

The Monte Carlo methods present an important drawback, inherent to their stochastic character: the Monte Carlo integration error has only a probabilistic bound. This means that it is not possible to assure that the error will not exceed a threshold, or in other words, Monte Carlo methods do not offer any guarantee that the expected accuracy is achieved in a concrete calculation. The analysis shows that a deterministic error bound can be established if deterministic values (instead of pseudo-random values) are used [40]. This is the basis of the *quasi-Monte Carlo methods*.

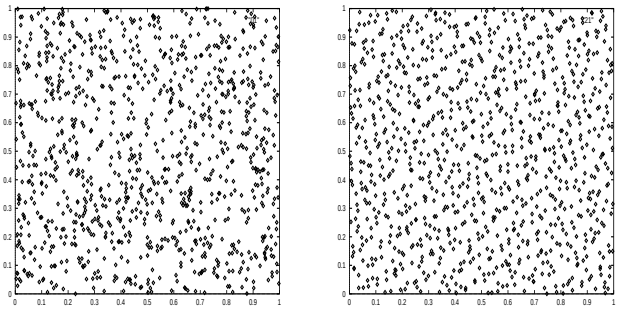
A quasi-Monte Carlo method can be seen as a deterministic version of a Monte Carlo method, in the sense that the random samples in the Monte Carlo method are replaced by well-chosen deterministic samples specially designed to be as much evenly distributed on the domain as possible (see Fig. 2.5). Thus, quasi-Monte Carlo integration follows the equation (2.31) but replacing Monte Carlo random samples by deterministic samples uniformly distributed on the domain. From here on, we will refer to these sequences of points as *quasi-Monte Carlo sequences*. Note that Monte Carlo integration and quasi-Monte Carlo integration, although based on the same formula (2.31), are conceptually different: in the first one we have a probabilistic error bound, so that it is possible to consider we are estimating an expected value and variance (see equation 2.32). The second one is a totally deterministic numerical method, so that we are not estimating any expected value, thus the variance has no sense in this context.

The only requirement for the deterministic sequence of points used in quasi-Monte Carlo is that such points are uniformly distributed on I^d in the sense of section 2.2.3.

If we consider integrands of finite variation, it can be proven [40] that quasi-Monte Carlo integration produces a deterministic error bound of $O(\frac{(logN)^d}{N})$ (d being the dimension) instead of the probabilistic error bound of $O(N^{-\frac{1}{2}})$

characteristic of Monte Carlo integration. Note that besides of the deterministic character of the error bound, the asymptotical behavior of the error is better when using quasi-Monte Carlo. This makes quasi-Monte Carlo integration to be superior to Monte Carlo integration in determinism and accuracy.

We can construct sequences of points uniformly distributed on I^d in the sense of section 2.2.3 (and thus valid to integrate on I^d) using both Monte Carlo and quasi-Monte Carlo generation. Quasi-Monte Carlo generation designs the sequences of points to be as evenly distributed as possible (or, in simple words, trying to fill empty spaces). This property is quantified by the *discrepancy*, that will be defined accurately in the next section. In Figure 2.5 right we can observe how the 2D points, that belong to a quasi-Monte Carlo sequence, appear to be more evenly distributed that the ones on Figure 2.5 left, generated by classic Monte Carlo. In terms of discrepancy, the quasi-Monte Carlo sequences have lower discrepancy than the Monte Carlo ones. This is the reason for them to be also known as *low discrepancy sequences*.



(a) (b)

Figure 2.5: *(a) Monte Carlo sampling (b) Quasi-Monte Carlo sampling (2-dimensional Halton points).*

As it will be seen with more detail in next section, the regularity (even distribution) of the samples on the integration domain happens to be more relevant for integration than true randomness [40]. This is the reason for quasi-Monte Carlo integration to perform better than Monte Carlo integration. Summarizing, we can affirm that the quasi-Monte Carlo method is superior to the Monte Carlo method not only in determinism but also in accuracy.

2.2.9 Discrepancy

The discrepancy [40] can be viewed as a quantitative measure for the deviation of a finite set of d-dimensional points from a totally even distribution (or, in other words, as a measure of the irregularity of the distribution). It is defined with respect to the family of subsets of I^d. Given a set of points $C = x_1, ..., x_n$ of I^d, we can define their star-discrepancy in the following way

$$Disc^*(C) = \underset{A \subset I^d}{max} \left| \frac{n(A)}{n} - V(A) \right| \tag{2.33}$$

where A is any box of I^d that contains the origin, $n(A)$ is the number of points that belong to box A and $V(A)$ is a normalized measure of the size of box A

(Figure 2.6). That is, the star-discrepancy is the maximum difference between the relative number of points of a box containing the origin and its relative size. Note that the star-discrepancy is only one of the formulations of the discrepancy. Other formulations according to the same basic idea can be found in [40].

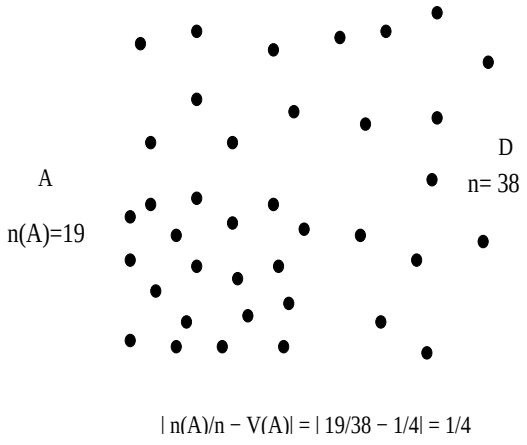


Figure 2.6: *The discrepancy is the maximum difference between the relative number of points and the relative size. In this example, the relative size of the box A is* $1/4$*, and its relative number of points is* $1/2$*.*

Looking again at Fig. 2.5, the set of points on (b), generated by quasi-Monte Carlo, seems to have a lower discrepancy that the ones on (a), generated by classic Monte Carlo. The effect we see is that points on Fig. 2.5 (b) have been generated trying to fill empty spaces. This is the basic idea of quasi-Monte Carlo generation.

It can be shown [40] that a set of N Monte Carlo generated points has a star-discrepancy $O(\sqrt{\frac{loglogN}{N}})$, whereas if we consider quasi-Monte Carlo sequences (like Halton or Weyl ones) the star-discrepancy behaves as $O(\frac{(logN)^d}{N})$, where d is the dimension of the points. In Fig. 2.7 we compare the asymptotical behavior as the dimension d grows. Note that, in quasi-Monte Carlo, the discrepancy grows as the dimension grows, but, since $N^{-1} < N^{-1/2}$, there is always a value of N from which on this discrepancy is lower than the Monte Carlo discrepancy.

Note that quasi-Monte Carlo sequences are specially designed to minimize the discrepancy. This is the reason for these sequences to be also known as *low discrepancy sequences.*

2.2.10 Discrepancy and uniform distribution in I^d

There is a close relation between discrepancy and uniform distribution. The two following properties are equivalent for an infinite sequence S of d-dimensional points in I^d [40]:

- S is uniformly distributed on I^d.
- $\lim_{N\to\infty} D^*(S_N) = 0$

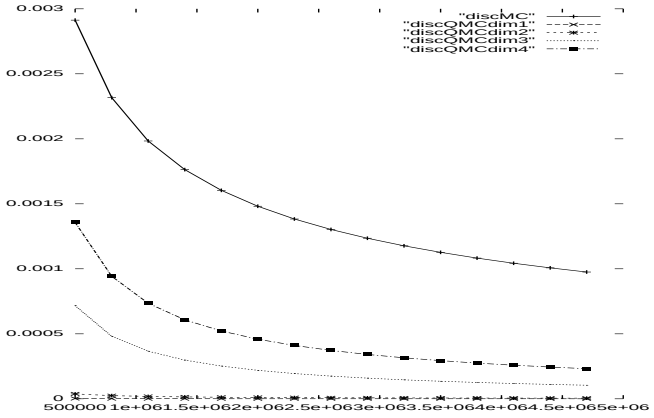


Figure 2.7: *Comparison of the asymptotical behavior of the discrepancy in Monte Carlo and quasi-Monte Carlo (dim. 1,2,3 and 4) generations. x-axis: N, y-axis: discrepancy*

where S_N are the first N points in S.

Note that from the discrepancies of the sets of points from Monte Carlo and quasi-Monte Carlo generation (see section 2.2.9) we can affirm that in both cases we are dealing with uniformly distributed sequences. However, we have also to note that, for a specific set of N points, the discrepancy is lower in quasi-Monte Carlo than in Monte-Carlo. This corresponds to the property of the points in quasi-Monte Carlo to be more evenly (more "uniformly") distributed on the domain.

2.2.11 Koksma-Hlawka inequality

The Koksma-Hlawka inequality (2.34) presents an upper bound for the absolute value of the error obtained in a Monte Carlo or quasi-Monte Carlo estimation. $D^*(x_1, ..., x_N)$ is the star-discrepancy of the set of points used, and $V(f)$ is the variation of the integrand in the sense of Hardy and Krause [40], a measure that describes the speed of change of the value of the integrand.

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_{I^s} f(u)du \right| \leq V(f)D^*(x_1, ..., x_N) \tag{2.34}$$

Note that if the integrand has finite $V(f)$, the convergence depends only on whether the star-discrepancy of the points tends to zero for the number of points tending to infinite. In other words, and according to section 2.2.10, the method converges, for integrands with finite variation, if the sequence of points x_k modulo I^d is uniformly distributed on I^d [40].

Continuous functions and also piecewise continuous one-dimensional functions happen to have finite $V(f)$, so that the previous assertion is valid for such integrand functions. This is not valid if the integrand has not finite variation (for instance, for piecewise continuous functions of higher dimensions).

The Koksma-Hlawka inequality presents two orthogonal ways of reduction of the error done in the integration:

- Reducing the variation of the function $V(f)$. This corresponds with the importance sampling techniques (see 2.2.7).
- Reducing the discrepancy of points $D^*(x_1, ..., x_N)$. This corresponds with the use of low discrepancy sequences, the basis of the quasi-Monte Carlo methods.

2.2.12 Discrepancy and convergence rate

From the Koksma-Hlawka inequality we have that, for finite variation functions, the integration error is bounded by the star-discrepancy of the samples. Since the star-discrepancy of the quasi-Monte Carlo sequences is known to be lower than the one of the Monte Carlo sequences (see section 2.2.9), we have a lower theoretical bound for the error in case of quasi-Monte Carlo integration. The exact behavior of the error in quasi-Monte Carlo integration is not known, but we know it is bounded by the discrepancy of the quasi-Monte Carlo sequences (for finite variation functions). Thus, this error is $O(\frac{(logN)^d}{N})$ (d being the dimension of the integral) in most quasi-Monte Carlo sequences. Note that this error decreases faster than the Monte Carlo error, that is known to be $O(N^{-0.5})$. Moreover, the quasi-Monte Carlo error is deterministic, whereas the Monte Carlo error is just probabilistic. This means that it is not possible to assure that the Monte Carlo error will not exceed a threshold.

This behavior is not theoretically justified for discontinuous integrands, like in case of the rendering equation (see 2.4) but, as we will see in chapter 3, it is extensible to general situations [64].

2.2.13 Quasi-Monte Carlo sequences

Most low discrepancy sequences (like Halton, Sobol or Weyl) are not ∞-uniform (note also that they fail most of the statistical tests for randomness of the pseudo-random numbers, see [40], chapter 7). They are just 1-uniform (see 2.2.2), contrary to Monte Carlo sequences. This means that there exists a correlation between a value of the sequence and the previous ones, and it makes impossible to obtain the sequence of uniformly distributed d-dimensional points needed for integration from 1 of these sequences (if $d > 1$). However, we can manage to obtain such a sequence of points using d independent 1-uniform quasi-Monte Carlo sequences. This is the procedure we will use in most cases [40].

Next we review the basic constructions of quasi-Monte Carlo sequences. More information about generating quasi-Monte Carlo sequences can be found in [40, 44, 32].

Radical inversion

The construction of some low discrepancy sequences is based on the radical inverse function [23]. Given a basis b, the radical inverse function Φ_b maps the natural numbers on the interval I. $\Phi_b(i)$ is defined then in this way:

$$\Phi_b(i) = \sum_{j=0}^{\infty} a_j(i) b^{-j-1} \tag{2.35}$$

where $a_j(i)$ is the j-th digit in the representation of i in basis b, that is:

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \tag{2.36}$$

For instance, let us consider basis 2. Since the representation of the number 19 in basis 2 is 10011, the radical inverse of 19 in basis 2, $\Phi_2(19)$, is equal to the value represented by 0.11001 in the same basis (that is, 1/2+1/4+1/32). Note that it corresponds to mirror at the decimal point the representation of the number in basis b. The sequence of numbers obtained in this way is known as the *Van der Corput* sequence of basis b.

This sequence is 1-uniform. Note that the algorithm of generation produces a single value in each interval of length 1/2, then in each interval of 1/4 and so on. Each interval will contain a sample before a second sample is placed. The discrepancy of this sequence is $O(\frac{logN}{N})$.

The Halton sequence

The Van der Corput sequences are 1-uniform, but not 2-uniform, because of the correlation between pairs of consecutive values. That means that a Van der Corput sequence is valid to integrate in dimension 1, but not in higher dimensions. As previously indicated, we can use d independent Van der Corput sequences to obtain an uniformly distributed sequence of d-dimensional points, valid to integrate in dimension d. Independency is guaranteed if we take as basis d numbers that are relative primes. This sequence is known as *Halton sequence*. Usually we consider as basis the first d prime numbers, although this is not the only possibility. So, the infinite Halton sequence in d dimensions can be built by

$$x_i = (\Phi_{p_1}(i), ..., \Phi_{p_d}(i)) \tag{2.37}$$

where $p_1, ..., p_d$ are the first d prime numbers.

For a set of N Halton points of dimension d we have [40] a star-discrepancy D^* of $O(\frac{(logN)^d}{N})$, that clearly improves the star-discrepancy of Monte Carlo generation. Note that this star-discrepancy converges to 0, so we can affirm that, as expected, the Halton sequence is uniformly distributed on its domain. According to Koksma-Hlawka inequality, this sequence produces, if the integrand has bounded variation, an integration error at most of the same order as the discrepancy, $O(\frac{(logN)^d}{N})$, improving dramatically the Monte Carlo error bound of $O(N^{-1/2})$. The minimum distance between 2 points in the set of N is at least $\frac{1}{N} \max_{1 \leq j \leq d} \frac{1}{p_j}$ (that is, the inverse of the lowest basis divided by the number of points) [40].

The Hammersley sequence

The Hammersley sequence is also based on radical inversion, and, as the Halton sequence, it is a sequence of d-dimensional points valid to integrate in dimension d. The main difference with the Halton sequence is that the number of samples has to be fixed *a priori*. Let N be this number of samples, we have

$$x_i = (i/N, \Phi_{p_1}(i), ..., \Phi_{p_{d-1}}(i)), i \in (0, ..., N-1) \tag{2.38}$$

where $p_1, ..., p_{d-1}$ are the first $d-1$ prime numbers, although it is enough to take relative primes.

The star-discrepancy of a set of N Hammersley points has a convergence rate of $O(\frac{(logN)^{d-1}}{N})$ that also improves the Monte Carlo generation. Following the same reasoning that in the Halton sequence, the integration error produced by the Hammersley sequence (if the integrand has bounded variation) is $O(\frac{(logN)^{d-1}}{N})$, improving dramatically the Monte Carlo rate. On the other hand, the minimum distance between 2 points in the set is at least $\frac{1}{N}$.

The Sobol sequence

The Sobol sequence is generated number-theoretically, rather than randomly, and successive points at any stage fill in the gaps in the previously generated distribution. Numbers in I are generated as binary fractions of length k bits from a set of k special binary fractions called direction numbers. In Sobol's original method, the jth number X_j is generated from a XOR (bitwise exclusive or) of the binary fractions V_i such that the ith bit of j is nonzero. Note that, as j increments, different values of the direction numbers flash in and out. The use of Sobol sequences leads to faster convergence compared to uniformly distributed random numbers, presenting a convergence rate $O(\frac{(logN)^d}{N})$, the same as Halton sequence.

The Weyl sequence

The Weyl sequence is originated from the fractional part of the multiples of the square root of prime numbers. That is

$$x_i = (frac(i\sqrt{p})) \tag{2.39}$$

where p is a prime number. The Weyl sequence is only 1-uniform, so we need to use, as in the case of the Halton sequence, d different Weyl sequences, d being the dimension of the domain over which we want to integrate. Traditionally the square roots of the first d prime numbers are used:

$$x_i = (frac(i\sqrt{p}_1), ..., frac(i\sqrt{p}_d)) \tag{2.40}$$

The discrepancy of a k-dimensional point sequence generated from d Weyl different sequences behaves as $O(\frac{(logN)^d}{N})$ (the same as with the Halton sequence), and this is the integration error order produced using Weyl sequences (if the integrand has finite variation).

∞-uniform sequences

There exist low discrepancy sequences that are supposedly ∞-uniform. That is, there is a conjecture that affirms that these sequences are ∞-uniform, but nobody has yet been able to prove it [33]. It implies that these sequences are supposedly valid to integrate in any dimension. For instance, the following sequence is supposed (but not proved yet) to be ∞-uniform:

$$x_i = frac(\pi^i) \tag{2.41}$$

where $frac(x)$ means the fractional part of x. π could be replaced with any transcendent number. Note that nobody has been able to prove that this kind of sequences are even 1-uniform, but the probability of these sequences to be ∞-uniform is equal to 1. This means that it is a safe bet that nobody in our lifetime will ever prove that $frac(\pi^i)$ is not ∞-uniform [33].

Scrambled sequences

The radical inverse function in basis b has subsequences of $b-1$ equidistant values spaced by $\frac{1}{b}$ that produce undesirable effects like alignement of points in lines. These effects, that are specially notable when using high basis in the case of Halton sequences, can be reduced by scrambling. Scrambling consists of changing the order of quasi-Monte Carlo sequences. One of the scrambling algorithms is the Faure's permutation [15]. This starts with the permutation (0,1) for basis $b = 2$, and it is extended to higher basis by distinguishing two cases:

- If b is even, we construct the permutation corresponding to basis b by multiplying by 2 the values of the permutation corresponding to $b/2$ and then appending the same values incremented by 1.
- If the basis b is odd, we take the permutation corresponding to $b-1$, incrementing by 1 each value greater or equal than $\frac{b-1}{2}$ and inserting in the middle the value $\frac{b-1}{2}$.

For instance, let us present the first Faure permutations:

- Basis 2: (0,1)
- Basis 3: (0,1,2)
- Basis 4: (0,2,1,3)
- Basis 5: (0,3,2,1,4)
- Basis 6: (0,2,4,1,3,5)
- Basis 7: (0,2,5,3,1,4,6)
- Basis 8: (0,4,2,6,1,5,3,7)
- ...

That means that, for instance, the scrambled Halton sequence of basis 4 is given by $(x_1, x_3, x_2, x_4, x_5, x_7, x_6, x_8, ...)$.

Other constructions than Faure's permutation have been used to obtain scrambled sequences, but always from the basic idea of changing the order of the quasi-Monte Carlo sequences [5, 61]. Scrambled sequences are particularly useful when dealing with a high Halton basis, because in this case the correlation is more notable.

2.3 Monte Carlo applied to radiosity

Monte Carlo methods are widely used in several areas of science like biology, chemistry, economy, etc. In this thesis we are interested in the application of Monte Carlo methods to computer graphics, and, specifically, to radiosity.

Monte Carlo methods have been widely used in the context of radiosity [52, 16, 47, 35, 38, 51, 4]. We have to distinguish between Monte Carlo algorithms that explicitely compute the form factors and Monte Carlo algorithms that do not compute the form factors. Note that, in the first case, once the form factors have been computed the radiosity system of equations must be solved, and, also, explicit computation of form factors presents $O(n^2)$ storage requirements, being n the number of patches in which the environment is discretized. The second kind of algorithms simulate the paths of the light particles or, in the case of progressive radiosity, compute at once only a row of form factors.

In both cases -computing or not form factors- Monte Carlo methods estimate the value of the radiosity integrals by generating random lines from suitable density functions. Random lines can be generated according to two different approaches [48, 2]:

- **Local approach.** Lines are cast in a local way, that is, they are cast from the surface of a given patch in the scene. These lines are referred to as **local lines**, and the simulations using such lines are referred to as **Local Monte Carlo**.
- **Global approach.** Lines are not related to a given patch, but they are related to the whole scene. Each line contributes to the simulation of several particle paths (or to the computation of several form factors). These lines are referred to as **global lines**, and the methods using such lines are referred to as **Global Monte Carlo**.

Note that the main difference between local and global lines is that in a local line approach we are only interested in the nearest intersection. Conversely, in a global line approach all intersections with the scene are considered, obtaining in this way an ordered list of intersections (see Fig. 2.8), so that each segment of the line is considered as a ray that leaves a surface and lands on another, doing the same job as lines in Local Monte Carlo.

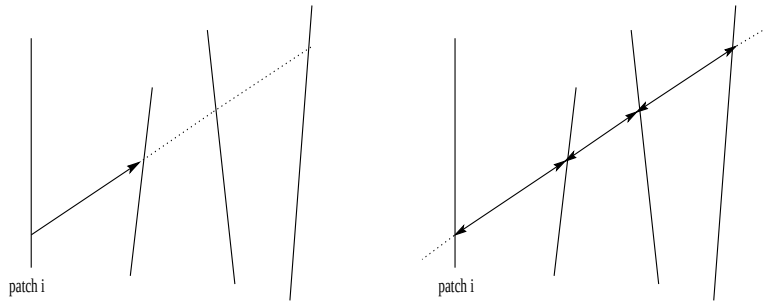


Figure 2.8: *(Left) Local (to patch i) lines. (Right) Global lines.*

In any case, the core of the Monte Carlo algorithms applied to radiosity is the Monte Carlo evaluation of the form factor integral. For this purpose we

need a suitable density of lines. Several approaches can be seen in sections 2.3.1, 2.3.2, 2.3.3 and 2.3.4. A complete exposition about this issue can be found in [48].

2.3.1 Monte Carlo evaluation of the form factor integral: local approaches

Our aim is to obtain a density of lines suitable for the estimation of the form factor integral. As seen previously, the Monte Carlo method needs a probability density function (pdf) $f(x)$ to generate the samples. The value of the integral becomes the expected value of the random variable $\frac{g(x)}{f(x)}$, and it can be estimated by generating N samples and computing their average. Next we consider three different approaches in which different pdf's have been used.

Area integral

We consider the form factor equation (2.5). Note that we are integrating over the areas of the patches. Monte Carlo integration needs a pdf (probability density function) to generate the samples. Using an uniform pdf $f_1(x,x') = \frac{1}{A_i A_j}$ corresponds to uniformly sample points x and x' over the surfaces of the patches i and j, respectively. For N such samples of pairs (x,x'), the form factor integral is approximated by (2.42). Note that no importance sampling is done.

$$F_{ij} \approx \frac{1}{N}\sum_{k=1}^{N}\frac{1}{\pi A_i}\frac{\frac{V(x,x')cos\theta_k cos\theta'_k}{r^2}}{\frac{1}{A_i A_j}} = \frac{A_j}{\pi N}\sum_{k=1}^{N}\frac{V(x,x')cos\theta_k cos\theta'_k}{r^2} \tag{2.42}$$

Thus, expression (2.42) allows to estimate form factor F_{ij} by sampling local lines between patch i and patch j.

Hemisphere integral

A second approach considers the hemisphere integral (equation (2.7)) instead of the area integral. Note that $V_j(x,\theta,\psi)$ indicates if the patch j is visible or not from point x in direction (θ,ψ). Taking again an uniform pdf $f_2(\theta,\psi,x) = \frac{1}{\pi^2 A_i}$, the Monte Carlo estimation of this integral is expressed in (2.43). This pdf corresponds to uniformly sampling a point x on patch i and a direction (θ,ϕ) in the hemisphere over patch i. That is, if ξ_1, ξ_2 are two random numbers obtained from an uniform distribution on $[0,1)$, (θ,ψ) can be obtained as $(\frac{\pi}{2}\xi_1, 2\pi\xi_2)$.

$$F_{ij} \approx \frac{1}{N}\sum_{k=1}^{N}\frac{1}{\pi A_i}\frac{V_{ij}(x_k,\theta_k,\phi_k)cos\theta_k sin\theta_k}{\frac{1}{\pi^2 A_i}} = \frac{\pi}{N}\sum_{k=1}^{N}V_j(x_k,\theta_k,\phi_k)cos\theta_k sin\theta_k \tag{2.43}$$

Note that this approach allows us to compute at once all the form factors from patch i by sampling local lines from this patch, but no importance sampling is done.

Hemisphere integral with importance sampling

Importance sampling consists of using probability density functions (pdf) that resemble the integrand. If we consider the hemisphere integral in equation (2.7), an appropriate pdf is $f_3(\theta, \psi, x) = \frac{cos\theta sin\theta}{\pi A_i}$. Now the Monte Carlo estimation of the integral is expressed in equation (2.44)

$$F_{ij} \approx \frac{1}{N}\sum_{k=1}^{N}\frac{1}{\pi A_i}\frac{V_j(x_k,\theta_k,\phi_k)cos\theta_k sin\theta_k}{\frac{cos\theta_k sin\theta_k}{\pi A_i}} = \frac{1}{N}\sum_{k=1}^{N}V_j(x_k,\theta_k,\psi_k) \qquad (2.44)$$

that is, the number of hits on patch j divided by the total number of samples. Note that in this way the form factor F_{ij} can be interpreted as the probability of a line that exiting patch i lands on patch j.

This pdf corresponds to the product of three independent terms:

$$f_3(\theta,\psi,x) = \frac{cos\theta sin\theta}{\pi A_i} = f_3^1(x)f_3^2(\psi)f_3^3(\theta) = \frac{1}{A_i}\times\frac{1}{2\pi}\times 2cos\theta sin\theta \qquad (2.45)$$

If we integrate these pdf's, we obtain the distribution functions according to which we sample the values. In the case of $\frac{1}{A_i}$, the distribution function corresponds to uniformly sampling a point on the area A_i. In the case of $\frac{1}{2\pi}$, it corresponds to uniformly sampling an angle ψ between 0 and 2π. In the last case, if we integrate $2cos\theta sin\theta$ we obtain $F(\theta) = sin^2\theta = 1 - cos^2\theta$. It corresponds to sampling angle θ from a sin^2 distribution (which in fact is equivalent to a cos^2 distribution). That is, $sin^2\theta$ is uniformly distributed, so we have to calculate $arcsin(\sqrt(\xi))$, where ξ is a random value obtained from an uniform distribution in $[0, 1)$.

2.3.2 Monte Carlo evaluation of the form factor integral: global approach

Form factors can also be estimated using a Monte Carlo global approach presented in [47, 50, 48]. This algorithm is based on *integral geometry*. Integral geometry allows us to establish an analogy between measures of sets of lines and form factors.

As seen in the previous section, form factor F_{ij} can be considered as the probability of a line that exiting patch i lands on patch j. In terms of measures of sets of lines, it can be considered as the quotient between the measure of the set of lines that cross both patches i and j and the measure of the set of lines that cross patch i. The method proposed in [47] is based on this fact, and uses Laplace's Rule to compute this probability, namely, the proportion of the lines that exiting patch i land in patch j. The difference with the third local approach (2.44) is that in this case global lines, instead of local lines, have been used, avoiding the waste of work typical of local approaches, in which only the closest intersection is employed: in the global approaches every intersection is employed. [48] shows that a global density of lines in the sense of integral geometry, that is, homogeneous and isotropic, submits on each patch the same density of lines obtained in the third local approach (see 2.3.1). This global

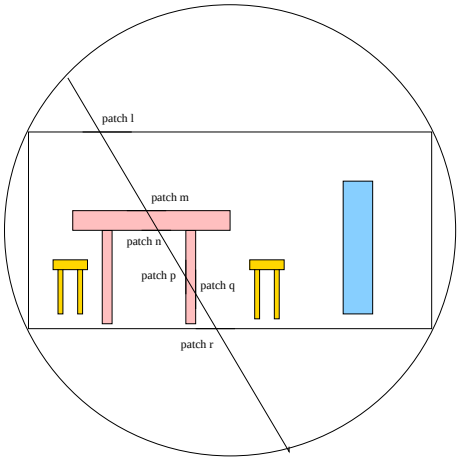


Figure 2.9: *The visibility pairs are in this case* (l, m), (n, p) *and* (q, r).

density of lines can be obtained in several ways, for instance sampling uniformly pairs of random points on the surface of a sphere that embodies the whole scene.

An estimator for the form factor F_{ij} can be computed in the following way. For each global line, we compute an ordered list of intersected patches. Then, we group the patches in the list of intersections in visibility pairs (figure 2.9). For every patch i in the scene, we have a counter of its number of intersections, r_i. For every pair of patches (i, j) we also have a counter of the number of lines that intersect both, r_{ij}. Then, the estimator of form factor F_{ij} is the ratio of the lines intersecting patch i that next intersect patch j:

$$F_{ij} \approx \frac{r_{ij}}{r_i} \tag{2.46}$$

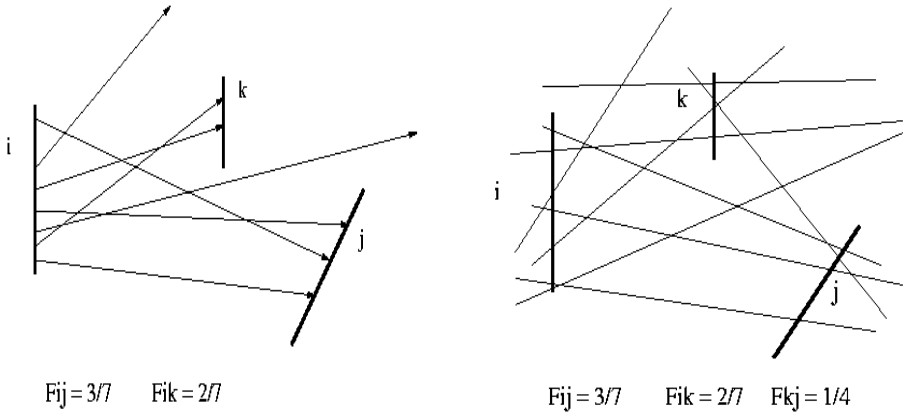
The estimation of form factors in both approaches (local and global) can be observed in Figure 2.10. Note that in the global approach (right) lines contribute to the estimation of the form factors from all patches, whereas in the local approach (left) lines contribute only to the estimation of the form factors from the patch where they are cast. This is the main difference between both approaches.

Another Monte Carlo approach to the estimation of form factors can be found in [42]. This is another global line based algorithm in which area projection is used.

2.3.3 Monte Carlo simulation of the light particles

Another kind of Monte Carlo radiosity algorithms simulates the trajectory of the light particles (photons) instead of computing first the form factors and then solving the system. Particle transport techniques were first used in Radiative Heat Transfer and introduced afterwards in the radiosity context. The first Monte Carlo radiosity approaches, like [52] and [41, 34] work in this sense. Note that these algorithms are in fact random walks [53, 24, 45, 28].

Pattanaik [41] presents a local approach that simulates the particle model of photons using random emission points and directions. Each particle bounces



(a) (b)

Figure 2.10: *(a) Form factors with local lines from patch i (b) Global density to compute form factors from all patches*

between the surfaces in the scene until absorbed. For each bounce the particle is either absorbed or rejected according to random sampling and the BRDF (Bidirectional Reflectance Distribution Function) of the reflection surface. If the particle is reflected, the outgoing photon flux of the surface is updated, and a new reflection direction has to be sampled. Note that the exiting point in the reflection surface is only sampled the first time, that is in the light sources. In the rest of bounces they use the intersection point as exiting point. The surface illumination is determined by finding the photon flux per area at different wavelengths. This *depth first* algorithm is valid not only for radiosity but also in the general global illumination context. A variation of this algorithm is when the exiting point can also be randomly obtained on each hit [48].

Another local Monte Carlo algorithm simulates the particle paths using a *breadth first* approach [16]. Each iteration of this algorithm corresponds to one bounce of all the light particles, that is, first iteration expands the primary power (first bounce), second iteration expands the second order power (second bounce) and so on. The process ends when the unshot power falls under a prefixed threshold. Each iteration must expand, for each surface in the scene, the unshot power by generating a number of random rays proportional to this unshot power. For each ray the exiting point and the outgoing direction must be sampled. There exists also other variants of local Monte Carlo particle tracing algorithms.

Note that in fact both depth first and breadth first algorithms correspond to the same simulation of the exiting power. In other words, they can be considered as the same algorithm. This algorithm obtains the random local directions in the same way that in the previously seen third local approach (2.44) to estimate form factors. This is, $(\theta, \psi) = (arcsin\sqrt{\xi_1}, 2\pi\xi_2)$, where (ξ_1, ξ_2) are both random numbers uniformly distributed in $[0, 1)$.

2.3.4 The Multipath method

The Multipath method, introduced in [51], is another Monte Carlo radiosity algorithm that simulates the trajectory of the light particles. Unlike the previous ones, the Multipath method uses global lines instead of local ones. It belongs to a family of methods that use random global lines (or directions) to transport energy, called by different authors global Monte Carlo, global radiosity or transillumination methods [49, 35, 63].

Global lines are independent on the surfaces or patches in the scene, in contraposition to local lines, used in the classic methods, which are dependent on the patches they are cast from. Global lines can take advantage not only from the closest intersection but also from all the intersections with the scene.

The Multipath method shows that it is possible to simulate a random walk by generating a global density of lines. We have to note that in the Multipath algorithm each light particle follows a path from state to state, the states being the patches in which the environment is discretized. So a particle in state i (patch i) will go to state j (patch j) according to a transition probability that is given by the form factor F_{ij}. So the density of lines has to be the same as the density of lines used to estimate the form factors in [47], that submits on each patch a distribution of exiting lines with the desired transition probabilities (form factors density). The global density of lines is obtained in the same way as in [47] (for instance by generating pairs of random points on the surface of a sphere that bounds the scene).

Note also that each global line will simulate the exchange of energy between several pairs of patches. In this way, every global line contributes to the advance of many simultaneous random paths (Fig. 2.11 a). So the name of Multipath is due to the fact that, at every moment, the state of the system can be interpreted as that of a scene with many paths, some of which are advanced simultaneously by each global line. We will next review the algorithm.

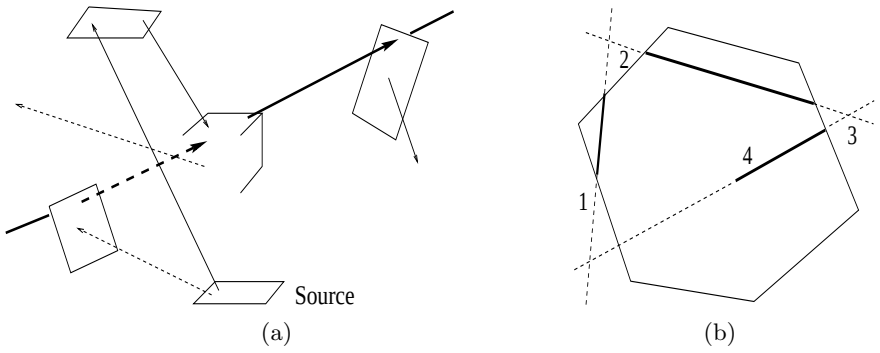


Figure 2.11: *(a) Multipath method. A global line (the thick one) simulates two paths, indicated with the continuous stroke and the dashed stroke (b) Multipath method. A path can contribute to the emission of power from several patches. In the figure, path 1-2-3-4 simulates paths 1-2-3-4, 2-3-4 and 3-4. This is similar to the covering paths [45].*

The Multipath algorithm

Global lines are intersected with the scene. For each line the intersections are sorted by distance in an intersection list. Each patch keeps two quantities. One records the power accumulated, the other one is the unshot power. For every pair of patches along the intersection list, they exchange their unshot power, decreased by the respective reflectances, and add to their accumulated power the unshot power of the other patch (also decreased by the reflectance). If a patch is a source, we keep also a third quantity, the emitted power per line exiting the source. Thus, if one of the patches of the pair is an emitter patch, we must add this emitted power per line to the unshot power. This "emitted per line" power is previously computed in the following way: given the number of lines we are going to cast, we compute for any light source patch beforehand the forecast number of lines passing through it. This number is, for a planar patch, proportional to the area of the patch [46]. The division of the total source power by this number gives the predicted power per line. In Fig. 2.12 there is the pseudocode of the algorithm. Table 2.1 describes the meaning of the variables used.

ρ_i	Reflectance of patch i
$\hat{B}_i$	Estimation of radiosity of patch i
POW_{ij}	Power transported from i to j
$ACCUM_i$	Accumulated outgoing power of patch i
$UNSHOT_i$	Unshot power of patch i
$EMIT_i$	Primary power of patch i
PPR_i	Primary power per line of patch i

Table 2.1: *Variables used in the Multipath algorithm.*

Note that $EMIT_i$ and PPR_i are only non-zero if i is a light source. On the other hand, $UNSHOT_i$ corresponds to the unshot power of patch i, that is, the power brought by the last line that has hit patch i, and that will go with next line hitting i. Note that this $UNSHOT_i$ simulates the distribution of non-primary power. Finally, PPR_i will be precomputed by dividing $EMIT_i$ by the forecast number of lines that will cross the patch, which for a planar patch is proportional to its area.

Advantages and drawbacks of Multipath method

The advantages of the Multipath method are the following. First, all intersections of a line with the scene are used. Second, the power transfer is bidirectional. Third, each random line contributes to several paths. And finally, each path is used to transport different logical paths (see Fig. 2.11b), as with the covering paths [45].

A drawback of the Multipath method is that in its first stages the distribution of power is only possible from light sources, and so most of the lines cast in these first stages (the lines that do not cross any light source) are wasted. To avoid this behavior a preprocess, called *first shot*, is done [6, 65]. In this preprocess the primary power is cast from the source patches by generating local lines that exit from the surface of each emitter patch. After that, the patches that have

for each patch i $ACCUM_i = UNSHOT_i = EMIT_i = PPR_i = 0$ **end for**
for each light source
    Initialize EMIT and PPR
end for
for each of the N lines to cast
    Generate line k
    Compute ordered list of intersected patches
    for each pair of patches i and j in the list
        $POW_{ij} = (UNSHOT_i + PPR_i) * \rho_j$
        $POW_{ji} = (UNSHOT_j + PPR_j) * \rho_i$
        $UNSHOT_i = POW_{ji}$; $UNSHOT_j = POW_{ij}$
        $ACCUM_i = ACCUM_i + POW_{ji}$; $ACCUM_j = ACCUM_j + POW_{ij}$
    end for
end for
for each patch i
    $\hat{B}_i = (ACCUM_i + EMIT_i)/A_i$
end for

Figure 2.12: Multipath algorithm. The variables are defined in Table 2.1

received some power will be the new sources instead of the original ones. Note that after this preprocess the power to be emitted is more widely distributed, decreasing the initial waste in global lines.

A detailed comparison of the Multipath method against the classic (local) methods can be found in [48].

2.4 Quasi-Monte Carlo applied to radiosity

Quasi-Monte Carlo techniques consist of using low discrepancy sequences in Monte Carlo integral (see 2.2.4). The application of low discrepancy sequences to the radiosity field was introduced in the nineties by Keller in [29], which presents a quasi-Monte Carlo algorithm that deals with diffuse and pure specular environments, obtaining a slightly better behavior with quasi-Monte Carlo sequences. The main idea of the algorithm is to calculate average local solutions of the radiosity equation for the scene elements A_k by the quasi random walk using a wavelet representation.

Another application of quasi-Monte Carlo generation is presented by Keller in [30, 32] to compute the form factors. The use of low discrepancy sequences in this context produces a better ray distribution in terms of discrepancy. That is, it increases the regularity of the samples, which is known to be more important than randomness for integration. The Monte Carlo rate of $O(N^{-\frac{1}{2}})$ is then outperformed with quasi-Monte Carlo integration. Different sequences, as Halton and Hammersley, were used.

The same author uses also quasi-Monte Carlo generation in a radiosity random walk (local approach) [31]. Low discrepancy sequences are superior to Monte Carlo random numbers when numerically calculating high-dimensional integrals. Halton sequences have been used in this quasi-random walk, where an equal absorption probability is used for all patches. This work also postulates

the advantage of using small Halton basis in front of higher values.

Other authors use quasi-Monte Carlo in the context of radiosity. L. Szirmay-Kalos et al. [63] apply low discrepancy sequences to the transillumination radiosity method, a Monte Carlo radiosity algorithm based on global parallel lines. Theoretical bounds have been given for the required number of samples in the numerical integration to guarantee convergence and to find the solution within a given accuracy. The method is efficient for environments with large and homogeneous faces. Scrambled Hammersley sequences have been used in this work to reduce the correlation between the points. In [62], a random walk is presented combining quasi-Monte Carlo and importance sampling.

A complete analysis of the quasi-Monte Carlo integration of the rendering equation is presented in [64], where they discuss about the discontinuity of the integrand. Since the integrand of the rendering equation is not continuous and so it does not have finite variation, it is not possible to apply the Koksma-Hlawka inequality in this context. This makes invalid the theoretical considerations that postulate the superiority of quasi-Monte Carlo in front of Monte Carlo integration. However, [64] shows that the error in quasi-Monte Carlo when integrating functions of non-finite variation will be between $O(N^{-(1-\epsilon)}) = O(\frac{(logN)^d}{N})$ and $O(N^{-\frac{1}{2}})$, approximating to this last rate when the dimension d of the integrand grows. That is, the error will not be worse than the Monte Carlo error in any case, but it approximates to it when the dimension grows. This corresponds in the rendering equation to the computation of higher bounces (note that the expansion of the rendering equation corresponds to a Neumann series containing a series of integrals of increasing dimension). [64] remarks that the higher order bounces, where the convergence of the quasi-Monte Carlo integration appears to be more degraded, have a lower contribution in the shooting random walk algorithms, specially when the mean reflectance of the environment is not very high. So the degradation of the convergence for higher dimensional integrals is compensated in part by their reduced contribution.

On the other hand, L.Neumann et al. [37] introduce a radiosity algorithm based on the idea of well distributed ray sets (WDRS). WDRS are sets of rays that connect mutually visible points and patches. Their construction is based on discrete importance sampling and it uses quasi-Monte Carlo sampling. The fact that the sampling is deterministic makes it possible and efficient the representation of WDRS. 4-dimensional Halton sequences with basis 2,7,3 and 5 have been used in this algorithm. This has been applied to complex scenes, where gains of half an order of magnitude are reported with respect to previous Monte Carlo radiosity algorithms [35]. However we note that the gains are not only due to the use of quasi-Monte Carlo integration but also to the new radiosity method.

Finally Bekaert et al. [3] make some interesting experiments that show that changes in the order of generation of points in (scrambled) quasi-Monte Carlo do not alter the convergence rate, but they have an important influence in the first stages of the simulation. Moreover this paper studies and compares different low discrepancy sequences, and it concludes that no sequence is clearly superior to the rest, so that the use of Halton sequence, the simplest one, is totally justified.

2.5 Conclusions

We have reviewed in this chapter the main topics involved in the present thesis. Since this thesis deals with radiosity methods, we have first examined the radiosity system of equations in the context of global illumination. Most of the work in this thesis is done in the sense of reducing the cost of Monte Carlo approaches to the radiosity method, so that this chapter also includes an introduction of the Monte Carlo method and a description of the use of different Monte Carlo techniques to solve for the radiosity of an environment, doing special incidence in the global line method Multipath.

In this thesis we deal with the quasi-Monte Carlo integration in the context of global Monte Carlo algorithms. Therefore, we have reviewed the main quasi-Monte Carlo concepts and also the existing applications of the quasi-Monte Carlo integration to the radiosity context.

Chapter 3

Use of quasi-Monte Carlo sequences in the Multipath method

3.1 Introduction

As seen in section 2.2.8, we call quasi-Monte Carlo sequences some deterministic sequences of numbers specially designed to be used in random simulations. The main feature of these sequences, that distinguishes them from ordinary Monte Carlo ones, is their lower discrepancy. As seen in section 2.2.9, discrepancy is a measure of the deviation of a set of values from the totally even (totally "uniform") distribution. Quasi-Monte Carlo sequences are also known as *low discrepancy sequences*. In other words, quasi-Monte Carlo samples are intended to be spread in a more even way over the domain.

Quasi-Monte Carlo sequences have been widely used in the last decades. They have been incorporated to Monte Carlo simulations, replacing the ordinary pseudo-random values in order to improve the performance of the simulation. We refer to this kind of simulations as *quasi-Monte Carlo methods*.

One of the fields in which quasi-Monte Carlo sequences have been introduced is realistic rendering, and particularly the radiosity context. Several of these applications have been reviewed in section 2.4. In the present chapter we present our study of the use of different quasi-Monte Carlo sequences in the Multipath algorithm, described in section 2.3.4. We study the improvement in Mean Square Error, the asymptotical behavior and also the influence of the line generation when using quasi-Monte Carlo.

The contributions presented in this chapter appear in [6, 9, 10].

3.2 Monte Carlo integration in the Multipath algorithm

The core of the radiosity problem is the computation of the form factors, which involves a double area integral, as seen in equation (2.5). Although the Multi-

path algorithm does not explicitly compute the form factors, the kernel of the problem is the same as when explicitely computing them. This means that the Multipath algorithm is a Monte Carlo algorithm that uses random lines to implicitly solve the form factor integral. The random lines that expand the light power in the Multipath algorithm have to be distributed according to the same density used in [47] to compute the form factors. Note that this density of lines corresponds to the importance sampling done in equation (2.44). The difference is that we use global lines instead of local ones. This density of lines can be obtained by different ways, for instance by sampling pairs of random points on the surface of a bounding sphere (like in [47]).

Another important point to deal with is the dimension of the integral. As seen in section 2.1.3, the form factor integral (equation 2.5) is a double area integral, so that it integrates area versus area (or area versus projected area). This means that the dimension of this integral is 4.

This has an important consequence in terms of quasi-Monte Carlo integration. As indicated in section 2.2.10, if we want to integrate over a 4-dimensional domain, we can use either one 4-uniform low discrepancy sequence or 4 independent 1-uniform low discrepancy sequences. In both cases we obtain a sequence of 4-dimensional points uniformly distributed in I^4. We have to remark that, in the second case, the 4 sequences have to be independent, namely correlations between them have to be avoided. Since low discrepancy sequences are 1-uniform, the question is to obtain 4 of these sequences that are independent between them. This can be obtained for instance by using 4 different basis that are relative prime in Van der Corput or Weyl generation. This corresponds, in terms of the Multipath algorithm, with the idea of avoiding correlations between the 4 values involved in the generation of each line.

3.3 Testing low discrepancy sequences in the Multipath algorithm

We call low discrepancy sequences -or quasi-Monte Carlo sequences- these deterministic sequences that are specially designed to minimize the discrepancy. As described in section 2.2.13, there are several ways to obtain this kind of sequences. Next we present the results obtained using different low discrepancy sequences in the context of the Multipath radiosity algorithm. We use the mean square error (MSE) (that is given by equation (3.1)) in relation to converged solutions, and we present also the images obtained with the different scenes we have tested. We compare the results in each case with the ones obtained using current Monte Carlo sequences.

$$MSE = \frac{\sum_i A_i * (\hat{B}_i - B_i)^2}{\sum_i A_i} \tag{3.1}$$

where A_i is the area of patch i, B_i is the estimation of the radiosity of patch i, and finally $\hat{B}_i$ corresponds to the exact value of this radiosity (we take this value from a converged solution).

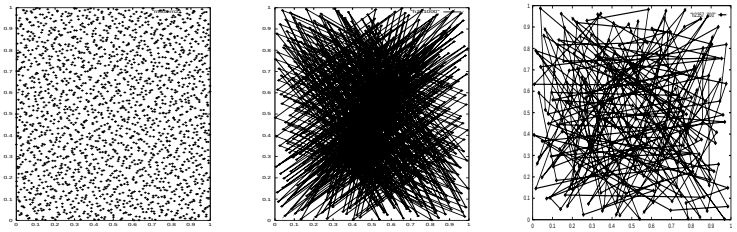
Two scenes have been tested. First, a simple scene in grey range -*SixCubes*- that represents a room with 6 cubes and a light source (see Fig. 3.8). Next, a more complex color scene -*Room*-, that consists of a room with a desk and a

table and chairs with some small objects on top, including a light source (see Fig. 3.9) [1]. All the executions have been done in a Pentium II at 350 Mhz.

3.3.1 Halton sequences

As described in section 2.2.13, a d-dimensional Halton sequence is a sequence of d-dimensional points in which each component comes from a different Van der Corput sequence. Taking as basis of the Van der Corput sequences the first k prime numbers, we guarantee the independence between each component. It allows us to obtain an uniform sequence of d-dimensional points valid to integrate in dimension d.

The dimension of the problem in the Multipath algorithm is given by the dimension of the form factor integral. Since it corresponds to a double area integral, the dimension is 4. Tests with Halton sequences of 2-dimensional points have been done, but the results have been deficient. This is due to the strong correlation between two consecutive 2-dimensional points, as shown in Fig. 3.1. Note the correlation in Fig. 3.1 (b): each point is "far" from the next one. From the dimension of the integral and the number of values needed to generate a random line, it seems logical to use a 4-dimensional Halton sequence in which the strong correlation between consecutive 2D points has disappeared, as shown in Fig. 3.1 (c). The discrepancy of this Halton sequence is $O(\frac{log^4 N}{N})$. Note that the sequence of 4-dimensional points generated in this way are uniformly distributed.



(a) (b) (c)

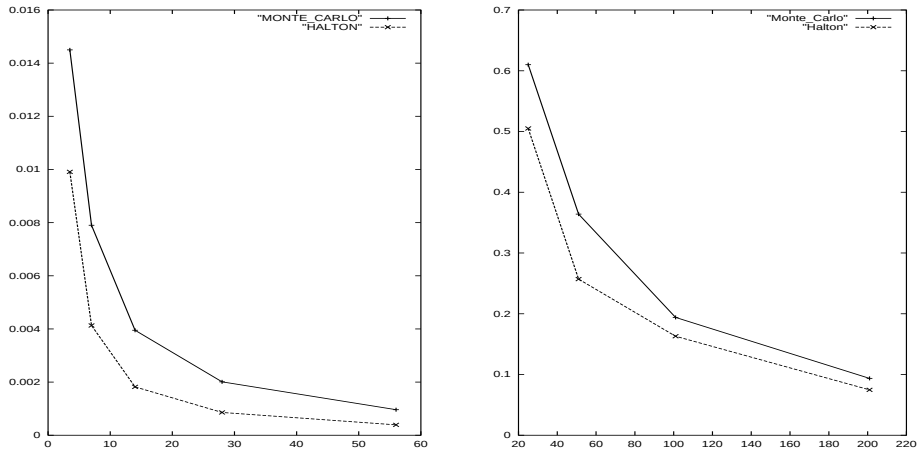
Figure 3.1: *(a) 2D points from a 2D Halton sequence are uniformly distributed in the unit square. They are valid to integrate on dimension 2. (b) Correlation between pairs of consecutive 2D Halton points: they are not valid to integrate on dimension 4. (c) Using a 4D Halton sequence: the correlation has disappeared. This values are valid to integrate on dimension 4.*

With respect to the basis, the independency is guaranteed if we take k numbers that are relative primes. In practice we take the first 4 prime numbers (2,3,5 and 7). Higher basis do not bring any additional gain, and in addition they slightly increase the cost of generation.

Two scenes have been tested. First, the scene *SixCubes*. A reduction of the MSE to nearly the half has been obtained (see Fig. 3.2 (a)). This gain is also observed in the resulting image, as it can be seen in Fig. 3.8 (b). Similar results have been obtained testing the scene *Room*, as shown in Fig. 3.2 (b) and in Fig. 3.9 (b). Note the better quality of the obtained images using quasi-Monte

[1] We will also test other scenes with quasi-Monte Carlo generation in the context of chapters 4 and 5

Carlo generation. Note also that the increase in computational cost due to the use of Halton sequences is negligible.



(a) (b)

Figure 3.2: *Halton generation. Graphs of time (sec.) vs. MSE. (a) Scene SixCubes. (b) Scene Room. Note in both cases the reduction of the MSE due to the use of the Halton sequences (basis 2,3,5 and 7)*

3.3.2 Hammersley sequences

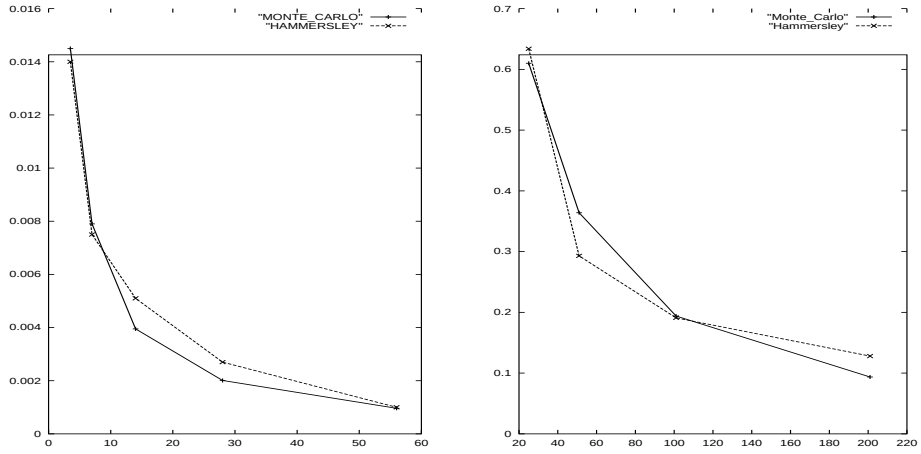
The Hammersley sequences are, like the Halton sequences, based on the radical inversion (Van der Corput sequence). The only difference with the Halton sequences is that the first component of the $i-th$ d-dimensional point is given by i/N, being N the total number of generated points. Note that this number of points has to be known in advance (conversely to the Halton generation), being a drawback of this generation.

We have used 4-dimensional Hammersley sequences, according to the discussion about the dimension done in the case of the Halton sequences. Note that here we only need 3 basis, having used 3, 5 and 7. The results, presented in Fig. 3.3, show that no gain is obtained with the use of the Hammersley sequences in front of classic Monte Carlo generation.

3.3.3 Weyl sequences

The Weyl sequences, described in section 2.2.13, are generated from the square roots of prime numbers. Like in the previous sequences, we need a 4-dimensional sequence to obtain a set of points valid to integrate in dimension 4. We have used the square roots of the first 4 prime numbers, namely 2, 3, 5 and 7. Note that in this way we work with 4 independent 1-uniform sequences that constitute a valid sequence of 4D points. Note that the discrepancy of such a sequence is $O(\frac{log^4 N}{N})$, the same as in the 4-dimensional Halton sequence.

We have also tested the scenes *SixCubes* and *Room*, comparing the results using Monte Carlo and Weyl generation. We can observe a reduction of the



(a) (b)

Figure 3.3: *Hammersley generation. Graphs of time (sec.) vs. MSE. (a) Scene SixCubes. (b) Scene Room. Note in both cases that no gain is obtained with the use of the Hammersley sequences*

MSE similar to the one obtained using Halton sequences (see Fig. 3.4). The only difference is that no reduction is observed when using a small number of lines. The images in Fig. 3.8 (c) and Fig. 3.9 (c) also show this behavior. No increase in cost is observed due to the use of the Weyl sequences.

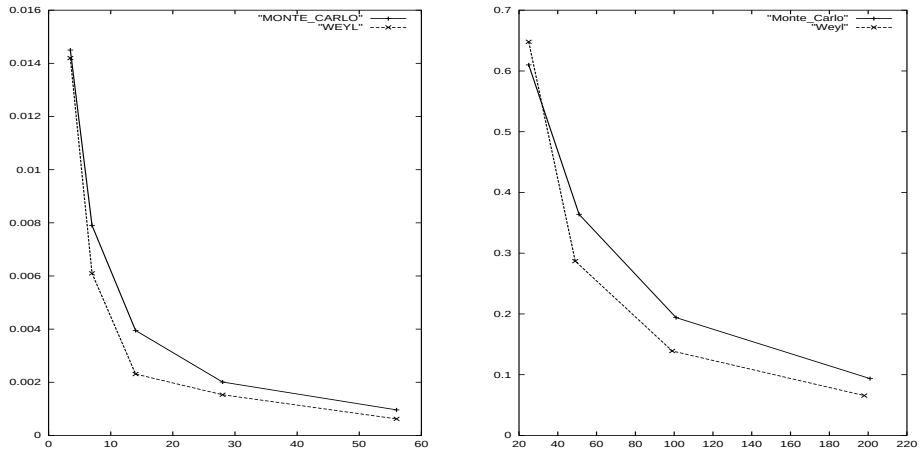
3.3.4 Sobol sequences

These quasi-Monte Carlo sequences, described in section 2.2.13, also produce a lower error compared to Monte Carlo numbers. As in the previous sequences, we have used a 4-dimensional sequence. The discrepancy of such a sequence is $O(\frac{log^4 N}{N})$, the same as in 4-dimensional Halton and Weyl sequences. We have tested the scene *Room*, and the gain obtained is similar to the gain in the case of Halton or Weyl sequences. This is shown in Fig. 3.5. In Fig. 3.8 (d) and Fig. 3.9 (d) we present the final images obtained using both Monte Carlo and Sobol sequences. Note the smoother final image obtained with the Sobol generation. Note also that the increase in computational cost due to Sobol generation is barely noticeable.

3.3.5 Scrambled sequences

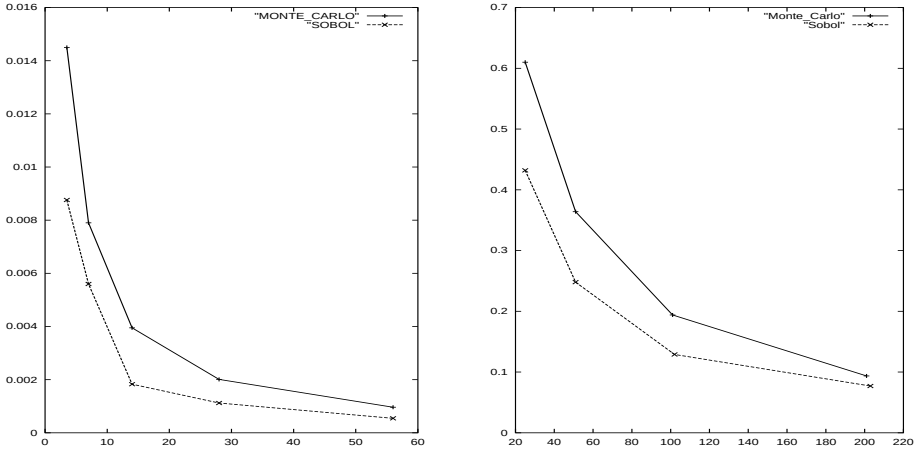
In some quasi-Monte Carlo sequences, like Halton, there is some correlation between the components of the sampled points. This correlation is reduced by the use of relative prime basis, but it is also noticeable when using high basis, appearing effects like alignment of points in lines. This phenomenon can be avoided using scrambled sequences. As indicated in section 2.2.13, scrambling consists of changing the order of the sampled values.

Scrambled Halton sequences have been tested in our scenes *Room* and *SixCubes*. No gain has been observed respect to the Halton sequences. This is



(a) (b)

Figure 3.4: *Weyl generation. Graphs of time (sec.) vs. MSE. (a) Scene Six-Cubes. (b) Scene Room. Note in both cases the reduction of the MSE due to the use of the Weyl sequence.*



(a) (b)

Figure 3.5: *Sobol generation. Graphs of time (sec.) vs. MSE. (a) Scene Six-Cubes. (b) Scene Room. Note in both cases the reduction of the MSE due to the use of the Sobol sequence.*

due to the fact that we only use small basis (2,3,5 and 7), and in this case the correlation is un-noticeable.

On the other hand, the nature of the scrambled Halton sequences makes impossible the generation of a value from the previous one. This produces a noticeable increase in computational cost in the generation of these values. This increase has repercussions on the total cost, producing in our tested scenes a total increase in cost of about 10 per cent. Note however that the more complex a scene is, the less will be the relative increase in cost.

The above remarks make scrambled sequences unattractive in the context of the Multipath algorithm. In Fig. 3.6 we have a graph that compares the performance of a Halton sequence vs. a scrambled Halton sequence. Note that no specific gain due to the use of scrambled sequences is observed.

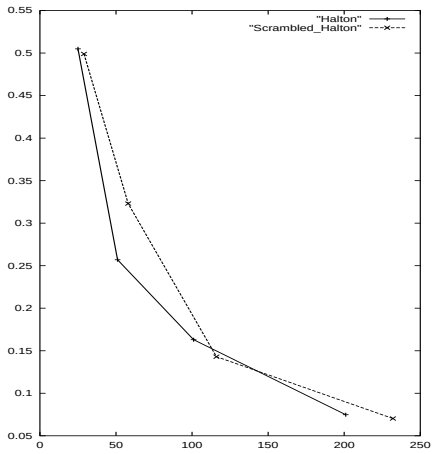


Figure 3.6: *Graph of time (sec.) vs. MSE for the scene Room. Note that the use of the scrambled Halton sequence does not bring any noticeable gain in front of the unscrambled Halton sequence.*

3.3.6 π^i modulo 1 sequence

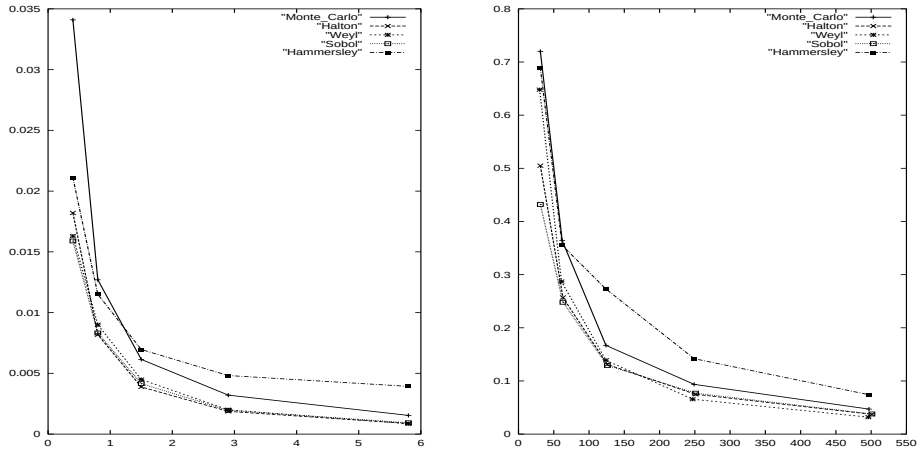
The sequence obtained from the fractional part of π^i (or π^i modulo 1) is considered ∞-uniform (see section 2.2.13). This kind of sequences, also known as completely uniformly distributed, are valid to integrate in any dimension. This means that from these sequences it is possible to obtain uniform sequences of k-dimensional points for any k.

However, we are strongly limited by the finite precision of computers. This means that we cannot represent all the decimals of π, but only a finite number of them. This fact makes patterns to appear in this sequence. Since we need a large number of random values, patterns make this sequence invalid for our purpose. Thus, this sequence is not useful in the context of our simulations.

3.3.7 Comparing the results

From the obtained results, we can conclude that the use of low discrepancy sequences improves the performance of the Multipath algorithm. This is due to

the higher regularity of the samples on the integration domain. However not all the quasi-Monte Carlo sequences yield a clear reduction of the error. The best results have been obtained using 4-dimensional Halton, Sobol and Weyl sequences. In these cases, reductions of the MSE to nearly the half have been obtained in our tests. In Fig. 3.7 we present a graph that compares the different quasi-Monte Carlo generations and the ordinary Monte Carlo generation for both scenes *Sixcubes* and *Room*.



(a) (b)

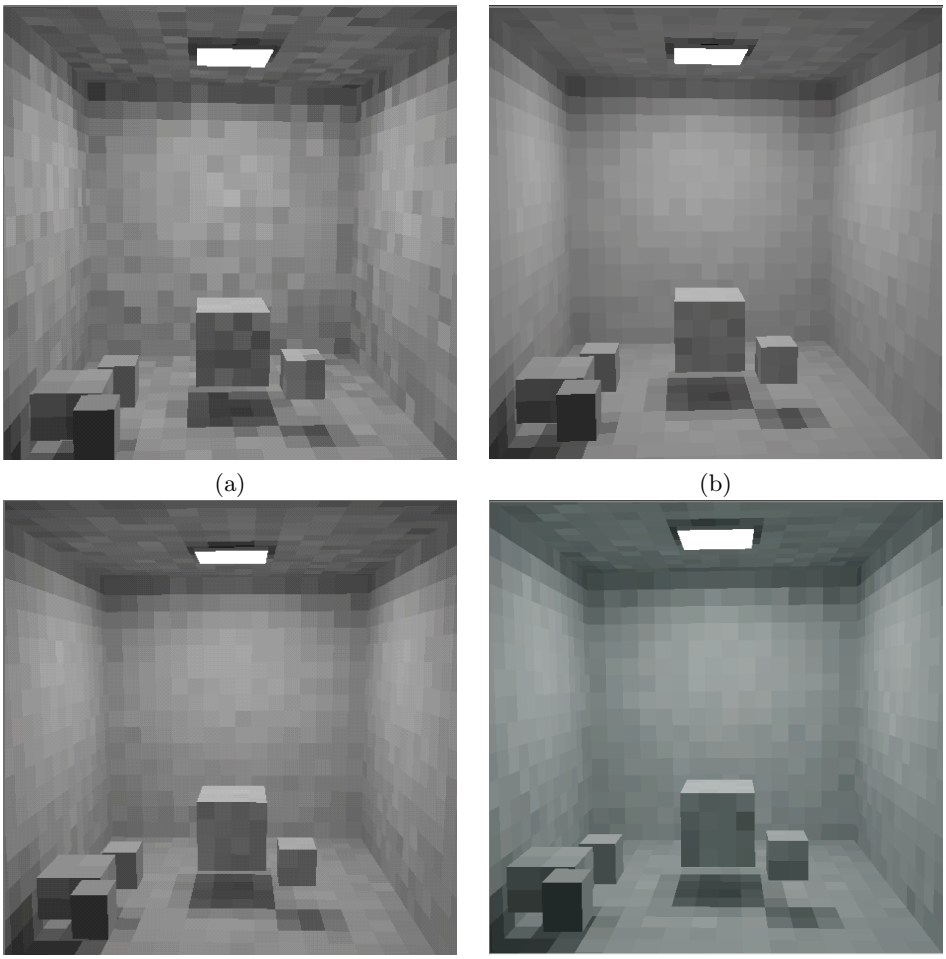
Figure 3.7: *Graphs of time (sec.) vs. MSE for the scene SixCubes (a) and Room (b). We compare the MSE obtained with the different low discrepancy sequences in front of the one obtained with Monte Carlo. The best behavior is obtained in both scenes with Halton, Sobol and Weyl sequences, that clearly improve the performance of the ordinary Monte Carlo generation.*

3.3.8 Asymptotical behavior

The expected value of the MSE decreases as $\frac{1}{N}$, being N the number of samples, in Monte Carlo integration. Then $log(MSE) \in O(log(N^{-1})$. That means that, if we take logarithms, the graph MSE vs. number of samples must be linear with slope -1.

We are interested in studying the asymptotical behavior in the case of using quasi-Monte Carlo sequences instead of Monte Carlo ones. The values of the experimental slope are presented in Table 3.1 for both scenes *SixCubes* and *Room*. This table shows that the convergence rate obtained using Halton, Sobol and Weyl sequences improves the one obtained with Monte Carlo sequences (slopes less than -1). On the other hand, the Hammersley sequence presents a worse convergence rate than Monte Carlo.

Finally we present the log-log graph for both scenes *SixCubes* (Fig. 3.10 (a)) and *Room* (Fig. 3.10 (b)). Note the best performance of Halton, Sobol and Weyl sequences in both scenes.

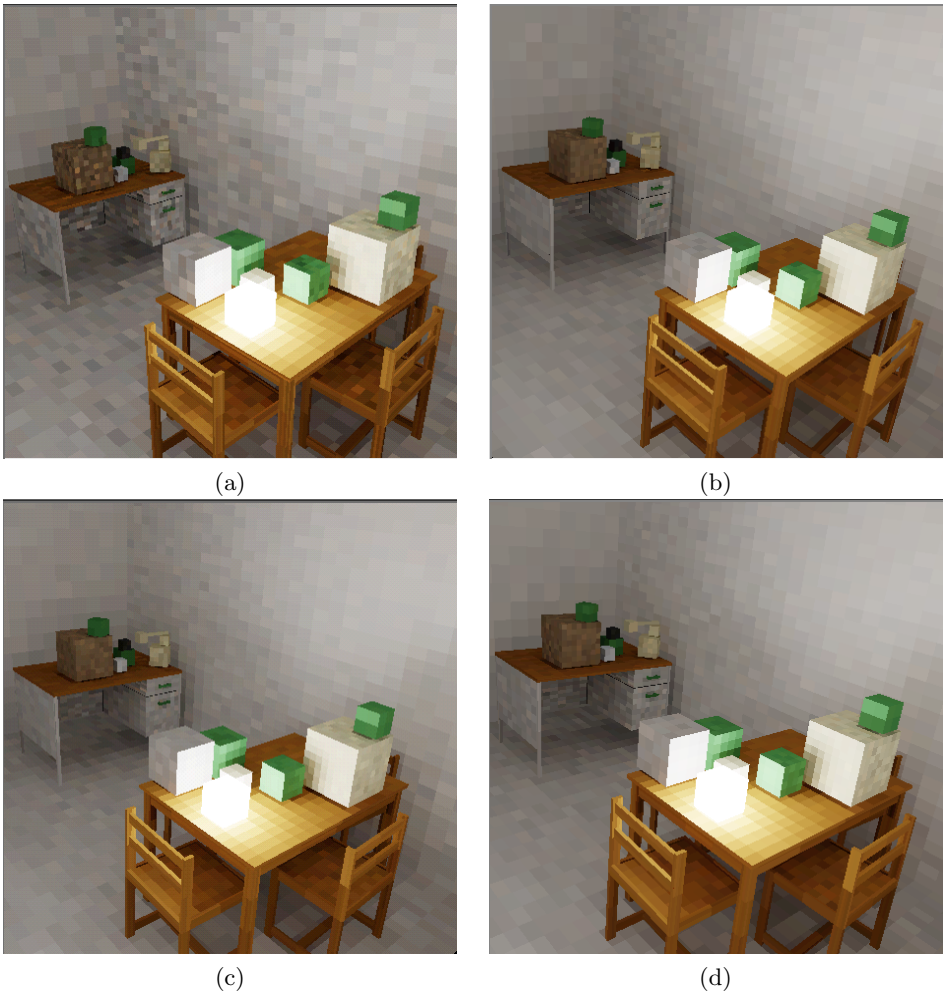


(a) (b) (c) (d)

Figure 3.8: *Scene SixCubes. (a) Monte Carlo generation. (b) Halton generation. (c) Weyl generation. (d) Sobol generation. Number of lines used in the four cases: 320 K (160 K local and 160 K global). Execution time: approx. 6 sec.*

SEQUENCE	*Scene SixCubes*	*Scene Room*
HALTON	-1.098	-1.113
SOBOL	-1.102	-1.122
HAMMERSLEY	-0.884	-0.874
WEYL	-1.157	-1.132

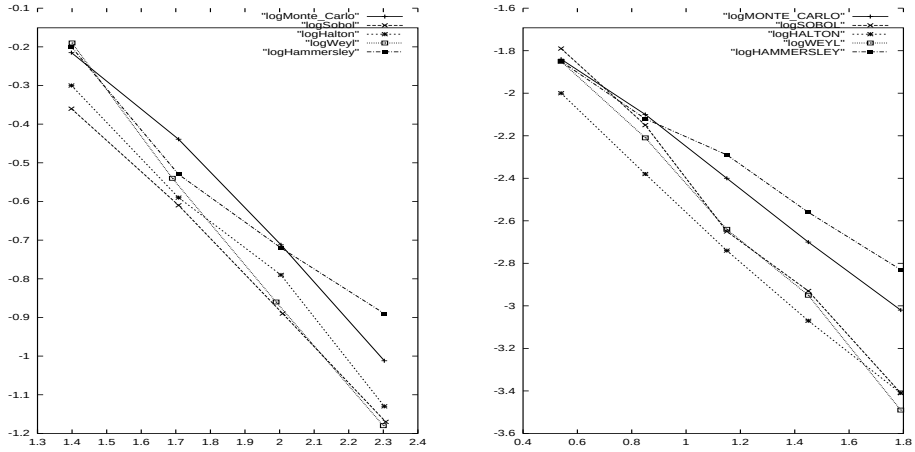
Table 3.1: *The slopes of the log(time) vs. log(MSE) show the asymptotical behavior. The Weyl sequence is slightly the best, whereas the Hammersley sequence is the worse.*



(a) (b)

(c) (d)

Figure 3.9: *Scene Room. (a) Monte Carlo generation. (b) Halton generation. (c) Weyl generation. (d) Sobol generation. Number of lines used in the four cases: 3456 K (1728 K local and 1728 K global). Execution time: approx. 200 sec.*



(a) (b)

Figure 3.10: *(a) Graph of log(time) vs. log(MSE) for scene SixCubes. (b) Graph of log(time) vs. log(MSE) for scene Room.*

3.4 Influence of the line generation

In this section we will deal with the influence on the results of the way global lines are generated. The global lines used in the Multipath algorithm have an uniform density in the sense of Integral Geometry, that is, homogeneous and isotropic. As seen in section 2.3.2, this density submits on each patch the same density of lines used in section 2.44 to estimate the form factors.

This global density of lines can be obtained in several ways. Now we will comment 4 of them [48, 46, 58].

3.4.1 Taking pairs of random points on a bounding sphere

In [46] it is shown that a density of global lines intersecting a convex body is given by $\frac{\cos\theta\cos\theta'}{r^2}d\sigma d\sigma'$, where θ, θ' are the angles of the intersecting line with the normals in the intersecting points, $d\sigma, d\sigma'$ are the area differentials in the same points and r is the length of the chord. If the convex body is a sphere (see Fig. 3.11a), the density becomes (save a constant factor) $d\sigma d\sigma'$. That is, taking pairs of uniform random points on the sphere surface we obtain a global uniform density of lines.

3.4.2 Lines from the walls of a convex bounding box

We can transform the density in 3.4.1 into $\cos\theta d\omega$. This new expression means that taking an uniform random point on the surface of the convex bounding box and a cosine weighted uniformly distributed direction (Fig. 3.11 b) we obtain the same global uniform density of lines as in 3.4.1. Since this result is valid for any convex bounding box, it is useful to use the bounding box of the scene (if convex) to generate the lines. An advantage of casting the lines from the walls

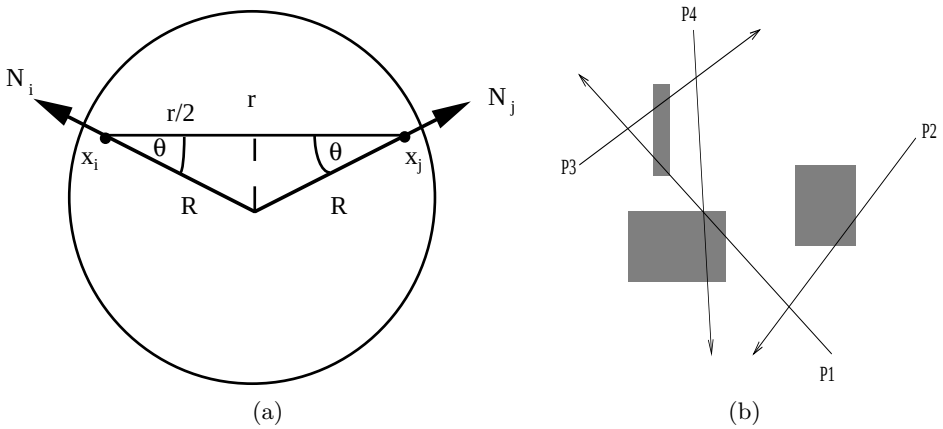


Figure 3.11: *(a) Random sampling from pairs of points on a sphere (b) Random sampling using lines from the walls in a convex bounding box*

in front of using a bounding sphere is that no lines are wasted, because all the lines intersect the scene.

3.4.3 Maximum circle

We sample an uniform random point on the surface of the bounding sphere (in fact, this is the same as sampling an uniform random direction). Then we take the circle orthogonal to this direction that contains the center of the sphere (that is, a maximum circle). Finally, we sample an uniform random point on this circle. With this point and the direction, we have the line (Fig. 3.12 (a). Note that this is equivalent to selecting a tangent plane (thus a point in the sphere) and a point in the projection of the sphere onto the plane.

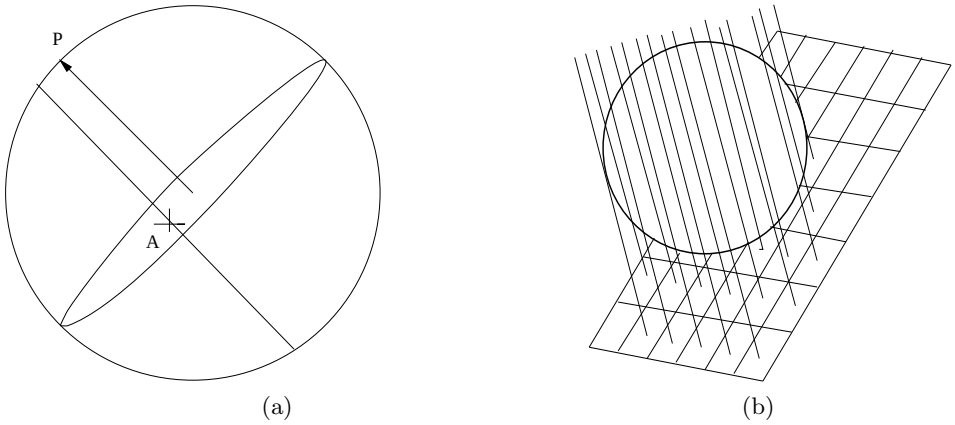


Figure 3.12: *(a) Random sampling using maximum circle (b) Random sampling using tangent planes*

METHOD	*SLOPE*
MONTE CARLO	-1.00
QMC: 2 POINTS ON THE SPHERE	-1.12
QMC: LOCAL FROM THE WALLS	-1.04
QMC: MAXIMUM CIRCLE	-1.02
QMC: TANGENT PLANES	-0.96

Table 3.2: *Scene 4CHAIRS. Asymptotical behavior of the MSE for Monte Carlo generation and the best of different quasi-Monte Carlo generations of global lines.*

3.4.4 Tangent planes: bundles of parallel lines

Another way to obtain random lines is using tangent planes. We have to sample a plane tangent to the sphere. To do this, we sample an uniform random point on the surface of the sphere, and then we construct the tangent plane at this point. Then we cast bundles of parallel lines orthogonal to the plane (Fig. 3.12 (b). Note that each tangent plane we generate produces a bundle of parallel lines. To avoid lines always passing by the center of the scene, the local origin in the plane is randomly jittered.

This technique offers several possibilities. An important point to consider is the balance between the number of planes (directions) and the number of lines per direction. On the other hand, the intersection of the lines with the scene can be accelerated in some ways, for instance by applying z-buffer techniques.

3.4.5 Comparing different line generations in the context of quasi-Monte Carlo

All these methods used to generate the random lines needed by the Multipath algorithm are equivalent in the sense that they produce the same density of lines when using classic Monte Carlo generation. Now we have studied their behavior when using low discrepancy sequences. Although the MSE obtained with each method are quite similar, the asymptotical behavior differs slightly. In Fig. 3.13 we have the graph log(n.lines) vs. log(MSE), and Table 3.2 presents the slopes for each case. Note that the best behavior is obtained when generating each line from 2 points on a bounding sphere. In this case Sobol sequences have resulted to be slightly better than the rest, contrary than in the other methods of generation, where Halton sequences were slightly superior. A scene with 4 chairs and a table (scene *4chairs*) has been used in this test.

3.5 Quasi-Monte Carlo and the high dimensionality of the Multipath integration

As we have seen in section 2.4, the integrand of the rendering equation is not continuous and so it does not have finite variation, not being possible to apply the Koksma-Hlawka inequality in this context. This makes invalid the theoretical considerations that postulate the superiority of quasi-Monte Carlo in

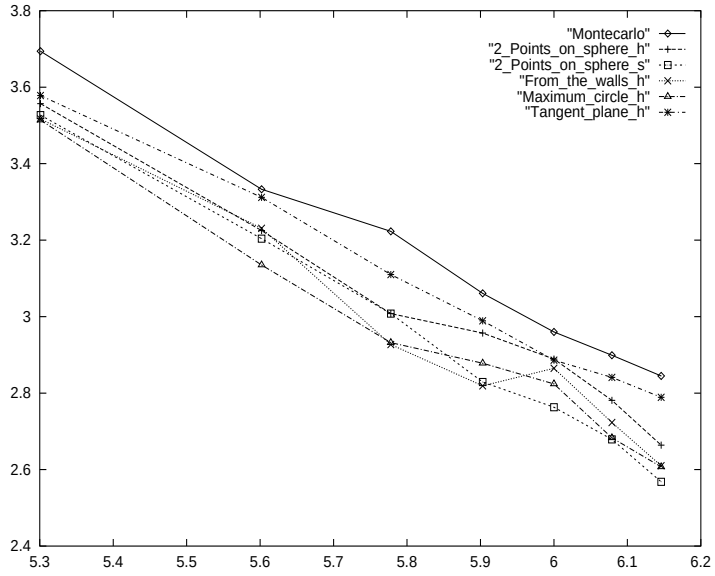


Figure 3.13: *Scene 4CHAIRS. Log-log graph. log(MSE) (in vertical axis) vs. log(number of lines) (horizontal axis).*

front of Monte Carlo integration. But, as presented in [64], the error of quasi-Monte Carlo when integrating functions of non-finite variation will be between $O(N^{-(1-\epsilon)}) = O(\frac{log^d N}{N})$ (d being the dimension) and $O(N^{-\frac{1}{2}})$, approximating to this last rate only when the dimension of the integrand grows. This corresponds, in the rendering equation, to the computation of higher bounces (note that the expansion of the rendering equation corresponds to a Neumann series containing a series of integrals of increasing dimension).

In the case of the Multipath algorithm, we have to note that each random line corresponds to different order bounces. Thus, it is not easy to distinguish the higher order bounces. Conversely, since we use a preprocess -first shot- [6, 65] to expand the direct illumination in which local lines are employed, it is easy to study the incidence of quasi-Monte Carlo in this first shot. We have done some experiments to establish the gain of quasi-Monte Carlo when considering only the direct illumination. We have used the scene *SixCubes*, and have compared Monte Carlo random generation with Halton generation. The results are shown in Fig. 3.14. MSE has been reduced between 50 and 60 per cent with the Halton sequences. Note that when we consider the complete simulation including non-direct illumination, the gain is between 40 and 50 per cent.

The important conclusion from these results is that the gain due to the use of quasi-Monte Carlo is more noticeable in the distribution of direct illumination than in higher order reflections. This difference is not due to the origin -local or global- of the lines, but to the fact that there is a loss of performance of the quasi-Monte Carlo integration when computing higher bounces, as indicated in

[64].

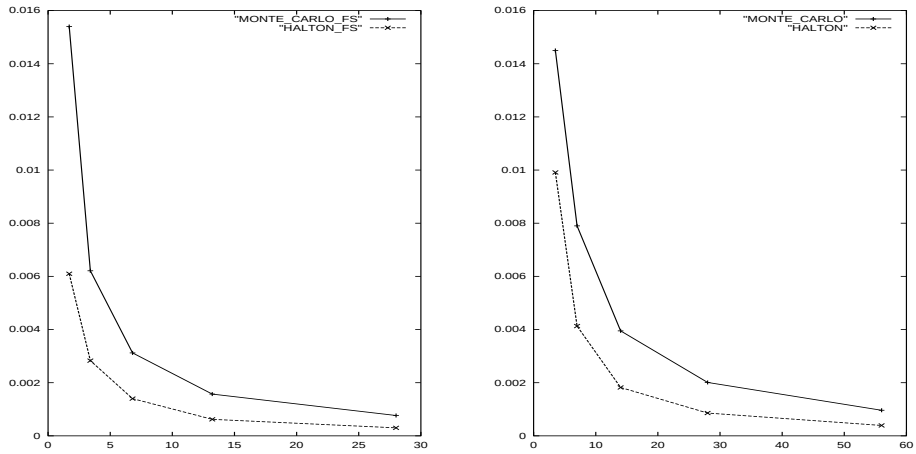


Figure 3.14: *Scene SixCubes. Graphs time vs. MSE for (a) Direct illumination. (b) Complete Multipath. Note that the gain obtained using quasi-Monte Carlo is more noticeable in direct illumination.*

3.6 Conclusions

This chapter has dealt with the use of quasi-Monte Carlo sequences in the context of the Multipath Monte Carlo radiosity algorithm. The main objective of this work has been to study the performance of quasi-Monte Carlo sequences in front of classical Monte Carlo pseudo-random numbers when used in the generation of the lines needed in the Multipath algorithm.

We have studied the performance of different quasi-Monte Carlo sequences. Some of these sequences, like Halton, Sobol and Weyl, appeared to be superior than classical Monte Carlo numbers in all our tests. Reductions of the mean square error to almost the half have been achieved at a similar time cost, and the asymptotical behavior of the MSE has also been better using these low discrepancy sequences. Other quasi-Monte Carlo sequences have not improved the performance of Monte Carlo: this is the case of the Hammersley sequence. We have also studied the scrambled Halton sequence, which produces no gain in relation with the ordinary Halton sequence. Finally, we have rejected the possibility of using ∞-uniform sequences based on powers of transcendent numbers, like π. The reason for this is the finite precision of the computers, that avoids considering true transcendent numbers.

On the other hand, we have studied the generation of the global uniform density of lines used in the Multipath algorithm. Different procedures to create this density have been presented. The main contribution in this sense is the study of the asymptotical behavior of the MSE when using these different procedures in the context of quasi-Monte Carlo. Some differences in this behavior have been established. Thus, the generation of the density of lines using pairs

of random points on the surface of a bounding sphere has appeared to have a slightly superior performance than the rest of the generations, specially when using Sobol sequences.

Finally we have studied the incidence of quasi-Monte Carlo sequences in the simulation of direct illumination, concluding, as expected, that the gain due to the use of quasi-Monte Carlo is clearly more noticeable in the distribution of direct illumination than in higher order reflections. This is due to the higher dimensionality of the integration in the case of indirect illumination.

Chapter 4

Hierarchical approach to the global Monte Carlo method for form factor computation

4.1 Introduction

We present in this chapter a hierarchical approach to the algorithm introduced in [47], and that has been described in section 2.3.2. This algorithm uses an uniform density of global lines that submits on each patch a density of lines suitable to estimate the form factors (see section 2.3.2). This density is uniform in the sense of Integral Geometry, that is, homogeneous and isotropic.

The strategy we propose is based on the subdivision of the scene in a hierarchy of sub-scenes, each one bounded by a sphere, and the generation, for each sphere, of a locally global density of lines that allows a more accurate estimation of the form factors. Note that most of the scenes we usually deal with are partially empty rooms. Thus, a big part of lines cast at the level of the whole scene only intersects the walls (and not the objects), contributing only to the estimation of form factors between the walls. Our new approach introduces, using the hierarchy of sub-scenes, densities of lines specific for the sub-scenes, that is, we cast more lines where they are more necessary, in the zones with a higher density of objects. This produces a more efficient use of the lines, resulting in a better performance of the new algorithm. Error in the estimation of the form factors has been notably reduced with our new approach.

The results in this chapter have been published in [8].

4.2 The hierarchical approach

4.2.1 Overview of the algorithm

For our hierarchical approach (HA) we will not only use a global sphere that wraps all the scene but also local spheres that bound one or some objects of

the scene. These local spheres allow to generate global lines in the same way as they are generated in [47], that is, from pairs of points sampled on the surface of the local sphere. These lines are valid in the context of the sub-scene bounded by the local sphere. That is, they constitute the uniform density described in 2.3.2 *in the context of the sub-scene*. In other words, these lines submit on each patch i in the sub-scene a density of lines suitable to estimate the form factors from i. We present in Fig. 4.1 a scheme of the algorithm.

Build hierarchy
Distribute lines among all the sub-scenes (how many in each sub-scene)
for each *sub-scene*
  for each *line to cast in the sub-scene*
    Generate line (from a pair of random points on the bounding sphere)
    Compute sorted list of intersected patches
    Update counters of intersections
  end for
end for
Estimate form factors

Figure 4.1: The algorithm.

4.2.2 Building the hierarchy

We build a binary tree of sub-scenes using the following naive clustering procedure. We first build the minimum bounding sphere for each single object in the scene. Then, we successively group pairs of spheres in a new bounding sphere. We always choose the pair of spheres with minimum distance between their centers. This process will be repeated until we have a single sphere which includes the whole scene (the main sphere) that constitutes the root of the binary tree. We note that, if there are k objects, the number of spheres created, including the root, is $2k$. Note also that the walls are not considered single objects. They only belong to the main sphere. Fig. 4.3 illustrates this method. A scheme of this clustering algorithm is presented in Fig. 4.2.

4.2.3 Establishing the number of lines per sub-scene

Fixing the percentage of lines to be cast in each sub-scene constitutes an heuristic question. In Fig. 4.4 we can see a sphere that contains two objects. A lot of the lines cast in it do not intersect any object in the sphere, so they are wasted. The emptier is the sphere, the bigger is the proportion of lines wasted.

We want to maximize the probability of a line to intersect some object in the sphere where it is cast. Let us consider the simplest case: a sphere with a simple object inside. In this case, Integral Geometry states that the probability of a line to intersect the object is given (for convex objects) by the quotient between the area of the object and the area of the sphere. For more than one object, this probability is not the sum of the single probabilities, but we can use it like a coarse approximation to the exact value.

for each *object in the scene*
Build its minimum bounding sphere
Mark the sphere as non-grouped
end for

while *exist two or more non-grouped spheres*
Group in a new (minimum) sphere the pair of non-grouped spheres with minimum distance between their centers
Mark new sphere as non-grouped
Mark both spheres as grouped
end while

Figure 4.2: The clustering algorithm.

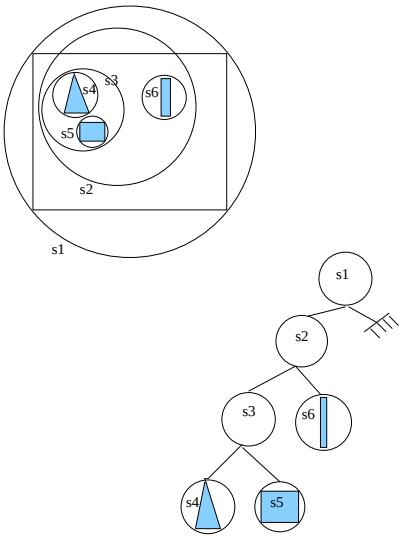


Figure 4.3: *Creation of the spheres hierarchy.*

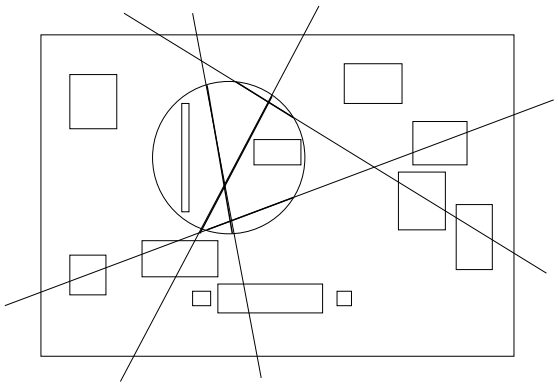


Figure 4.4: *Waste of lines in a local sphere.*

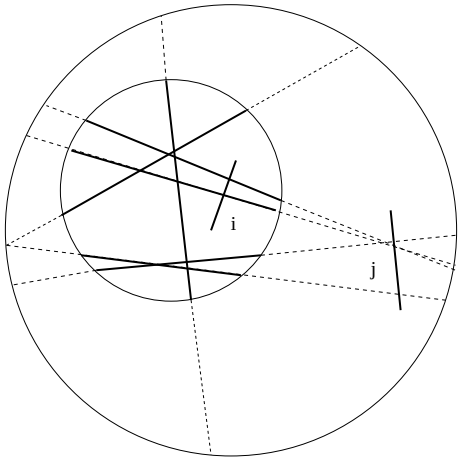


Figure 4.5: *Lines cast in the internal sphere are valid to estimate form factors from patch i, but not from patch j.*

Thus, it seems reasonable to distribute the number of lines to cast according to these probabilities. Thus the percentage of lines to cast in a sphere will be proportional to the quotient between the sum of the areas of its inner objects and the area of the sphere. The main sphere has been considered apart in our implementation, since it contains the walls. So we cast a fixed percentage of lines in this sphere. In our tests we have obtained best performance casting a 20 per cent in the main sphere, and distributing the remaining 80 per cent between the sub-scenes, according to the previous criterion.

4.2.4 Form factors estimation

One of the main questions in our new algorithm concerns the form factor estimation. From [47] we have that form factor F_{ij} can be considered as the probability of a line that leaves patch i to reach patch j, and so it can be estimated, according to Laplace's Rule, as the proportion of lines crossing patch i that next hit patch j. In our new hierarchical approach, different densities of lines are generated in the different spheres. Lines cast in sphere S submerge patches into this sphere in an uniform density valid to estimate form factors from these patches. This is not valid for patches out of sphere S, because there this density of lines is not uniform. From here we have that to estimate F_{ij} we can use lines cast in spheres that contain patch i, because these lines submerge patch i in an uniform density; but conversely we can not use lines cast in spheres that do not contain patch i. See for example Fig. 4.5. In it, the density of lines cast in the external sphere is uniform for both patches i and j. But the distribution of lines cast in the internal sphere is only uniform for patch i, that belongs to this sphere, but not for patch j.

We have to consider the following intersection counters (the same as in the classic method [47]):

- For each patch i, r_i is the number of lines that intersect it.

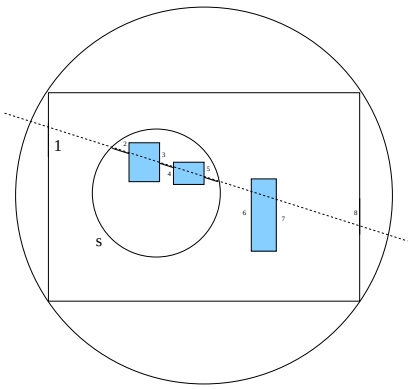


Figure 4.6: *Line cast in a sub-scene bounded by sphere S.*

- For each pair of patches i and j, r_{ij} is the number of lines that intersect patch i and next intersection is patch j.

For each line, we obtain the sorted list of intersected patches. We group in pairs the list. For every pair (i, j) in the list, we will proceed as follows:

- If both patches i and j belong to the sphere where the line has been cast, we increase the counters r_i, r_{ij}, r_j and r_{ji}.
- If only one patch, for instance i, belongs to the sphere where the line has been cast, we only increase r_i and r_{ij}.
- If neither of the two patches belong to the sphere where the line has been cast, we do not increase any counter.

Fig. 4.6 illustrates this procedure. The line has been cast in sphere S. So, the pair of patches $7-8$ does not bear any modification to the counters, because neither of these patches belong to the sphere. Conversely, since patches 3 and 4 are both into S, the pair $3-4$ increases all the involved counters. Finally, pairs $1-2$ and $5-6$ only modify the counters relative to the patches 2 and 5, those that belong to sphere S.

Note that the following identity is satisfied

$$r_i = \sum_{\forall j} r_{ij}$$

Form factor F_{ij} can be estimated in the same way as in the classical method [47]. The following estimator

$$\hat{F}_{ij}^{1} = \frac{r_{ij}}{r_i} \tag{4.1}$$

comes from considering Laplace's Rule. But we can also consider other estimators. From the reciprocity relation (4.2), we can obtain F_{ij} from F_{ji}. Note that, since the density of lines is not the same for every patch, we have that r_{ij} is not necessarily the same as r_{ji}. Using reciprocity equation we establish a new estimator, presented in equation (4.3).

N.OF LINES	ERROR $\hat{F}_{ij}^1$	ERROR $\hat{F}_{ij}^3$	ERROR $\hat{F}_{ij}^4$
$5 * 10^4$	1.1796	8.82378	1.1224
10^5	0.60453	4.27148	0.50391
$5 * 10^5$	0.11347	0.81667	0.09735
10^6	0.05634	0.44048	0.04850

Table 4.1: *Comparison of the error made using different estimators (scene NINECUBES).*

$$F_{ij} = \frac{A_j}{A_i} F_{ji} \tag{4.2}$$

$$\hat{F}_{ij}^2 = \frac{A_j}{A_i} \frac{r_{ji}}{r_j} \tag{4.3}$$

Moreover, we can consider estimators obtained by weighting of $\hat{F}_{ij}^1$ and $\hat{F}_{ij}^2$. For instance, using as weights of every estimator the number of lines intersecting each patch, r_i and r_j, we obtain

$$\hat{F}_{ij}^3 = \frac{r_i \frac{r_{ij}}{r_i} + r_j \frac{A_j}{A_i} \frac{r_{ji}}{r_j}}{r_i + r_j} \tag{4.4}$$

Note that the weight r_i only depends on patch i and the weight r_j only depends on patch j. They are not depending on the relation between both patches. If we use as weights the number of lines that leaves a patch and lands on the other one, namely r_{ij} and r_{ji}, then we have weights that are related at the same time with both patches. We have then the following estimator

$$\hat{F}_{ij}^4 = \frac{r_{ij} \frac{r_{ij}}{r_i} + r_{ji} \frac{A_j}{A_i} \frac{r_{ji}}{r_j}}{r_{ij} + r_{ji}} \tag{4.5}$$

An empirical comparison of the results obtained with different estimators (see Table 4.1) shows that $\hat{F}_{ij}^4$ presents the best behavior, but the simplest $\hat{F}_{ij}^1$ has also a good performance. We have used in this comparison the form factor error in scene NINECUBES.

4.3 Analysis of results

We present here a comparison between the results obtained using both classical approach [47] and our new approach. We have compared the accuracy of the estimated form factors. We should compare these form factors with an exact analytical solution. Since it is not possible to obtain this, we have used a converged solution obtained with the first Monte Carlo local method in section 2.3.1, casting a large number of lines (approx. 10000) for each pair of patches.

4.3.1 Error in form factors

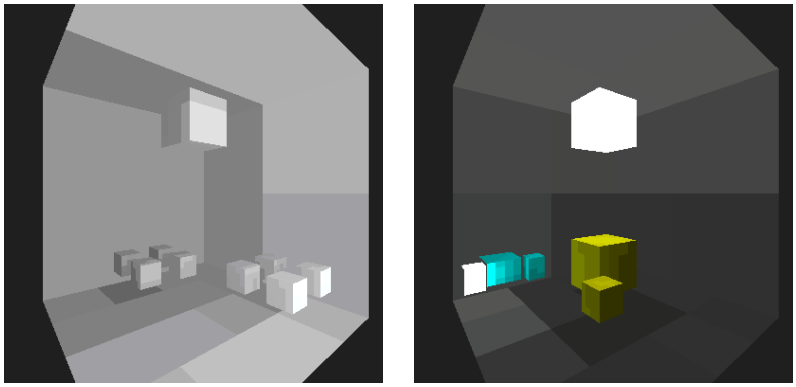
We define form factor error (FFE) in the following way:

$$FFE = \sum_{\forall i,j} (\hat{F_{ij}} - F_{ij})^2 \tag{4.6}$$

where $\hat{F_{ij}}$ is the estimate value of the form factor and F_{ij} is the exact value. Note that all form factors have been given the same weight in this expression.

4.3.2 Results

We have chosen for our tests two simple scenes. The first one, shown in Fig. 4.7 (a), is composed of 9 cubes inside a cubic room, presented in 2 groups of 4. The second one, shown in Fig. 4.7 (b), is composed by 6 cubes.



(a) (b)

Figure 4.7: *(a) Scene NINECUBES. (b) Scene SIXCUBES.*

We have considered, for each execution, the number of cast lines, the execution time and the form factor error (FFE). Next we present the results obtained for both scenes, in which we compare the classical algorithm (CA) and the new hierarchical approach (HA). We have used the estimator in (4.1). All the executions have been done in a Pentium II at 350 Mhz.

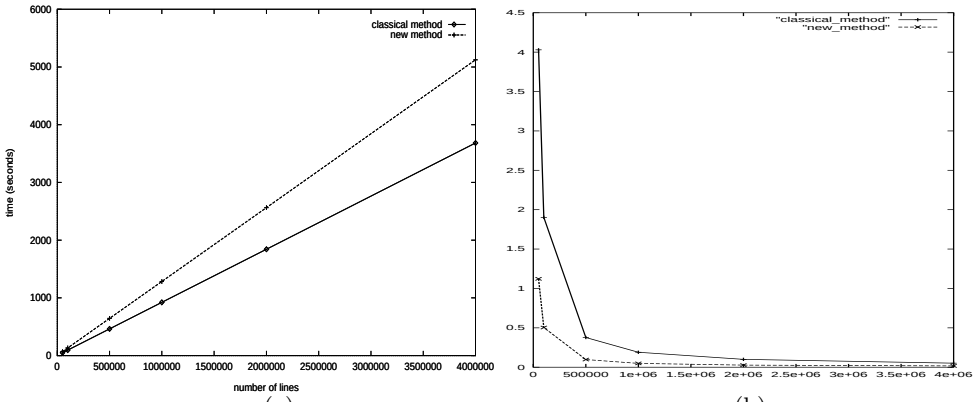
Scene NINECUBES

We present the results obtained with this scene in Table 4.2. In this table we compare, for the same number of lines, the execution time and the form factors error for both the classical method and the new approach. Note that although execution time is, for the same number of lines, higher in the new approach in approximately a 40 per cent (due to the higher cost of lines cast in the sub-scenes), this is widely compensated for the reduction in the FFE. This is reflected by relative efficiency, which is obtained as the quotient of the respective products of time and FFE, and shows the speed-up factor of the new algorithm in front of the classical one. Note that this value ranges from 2.5 to 3.

LINES	TIME CL.	FFE CL.	TIME HI.	FFE HI.	REL.EFFIC.
$5 * 10^4$	49	4.0294	68	1.1224	2.59
10^5	95	1.9018	132	0.5039	2.72
$5 * 10^5$	463	0.37706	644	0.09735	2.78
10^6	923	0.18986	1284	0.04850	2.82
$2 * 10^6$	1843	0.09987	2564	0.02690	2.66

Table 4.2: *Results for the scene NINECUBES. Second and fourth column show execution time for classic and hierarchical approach, respectively. Third and fifth column show the error for both approaches. Last column shows the relative efficiency of the hierarchical approach respect to the classic approach.*

We also present graphs of number of lines vs. time (Fig. 4.8 (a)), number of lines vs. FFE (Fig. 4.8 (b)) and finally time vs. FFE (Fig. 4.9) for both algorithms. Last graph expresses the superiority of the new approach.



(a) (b)

Figure 4.8: *Scene NINECUBES. (a) Number of lines (x axis) vs. time (y axis). (b) Number of lines (x axis) vs. FFE (y axis). We compare the results of classic method with the new hierarchical method.*

Scene SIXCUBES

Similar results have been obtained for scene SIXCUBES. These results are presented in Table 4.3, where it is possible to compare the performance of the new approach in front of the classical method. Relative efficiency shows the speed-up factor of the new approach in front of the classical method. Note that this value is, in most cases, between 2 and 3. As in the previous scene, the increase in cost appears to be clearly compensated by the reduction in the FFE.

We present graphs of number of lines vs. time (Fig. 4.10 (a)), number of lines vs. FFE (Fig. 4.10 (b)) and finally time vs. FFE (Fig. 4.11) for both algorithms. Last graph expresses the superiority of the new approach.

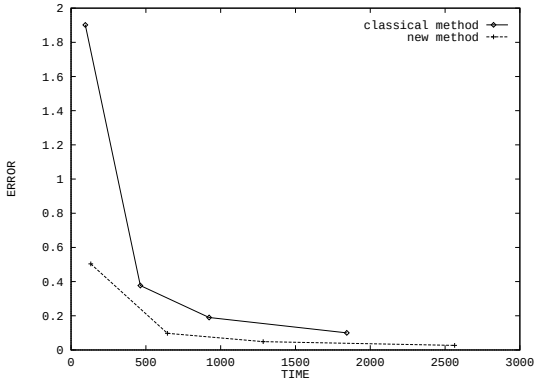
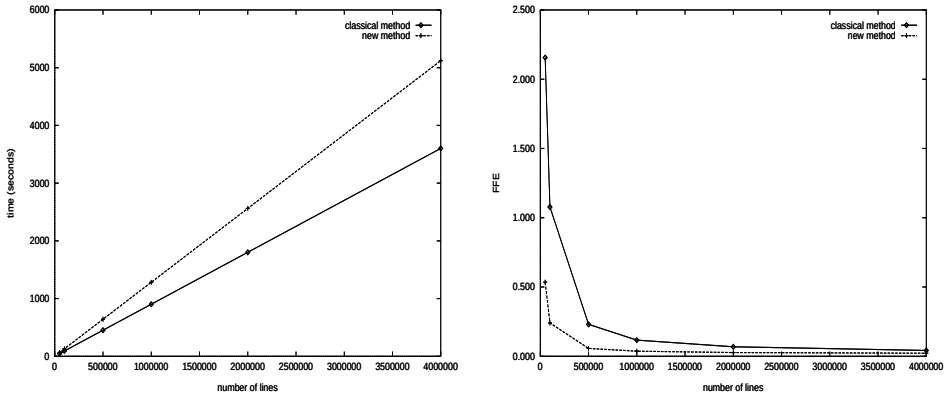


Figure 4.9: *Scene NINECUBES. Time (x axis) vs. FFE (y axis) for classic (continuous line) and new hierarchical (dashed line) method.*

LINES	TIME CL.	FFE CL.	TIME HI.	FFE HI.	REL.EFFIC.
$5 * 10^4$	47	2.1563	66	0.53527	2.87
10^5	92	1.0776	130	0.24104	3.17
$5 * 10^5$	452	0.23079	642	0.05680	2.84
10^6	902	0.11680	1282	0.03704	2.28
$2 * 10^6$	1802	0.06892	2562	0.02703	1.79

Table 4.3: *Results for the scene SIXCUBES. Second and fourth column show time for classic and hierarchical approach, respectively. Third and fifth column show the error for both approaches. Last column shows the relative efficiency of the hierarchical approach respect to the classic approach.*



(a) (b)

Figure 4.10: *Scene SIXCUBES. (a) Number of lines (x axis) vs. time (y axis). (b) Number of lines (x axis) vs. FFE (y axis). We compare classic method with new hierarchical method.*

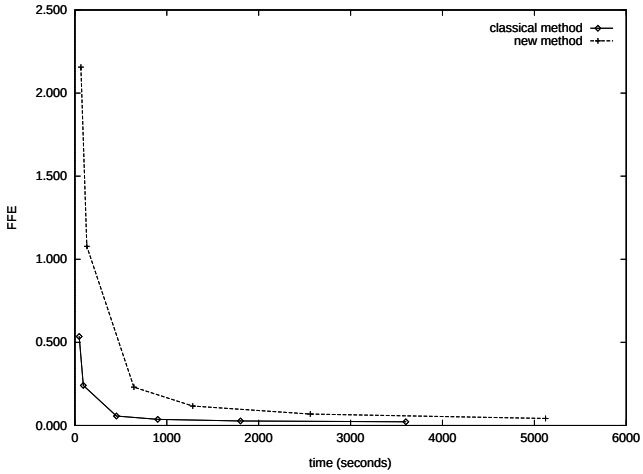


Figure 4.11: *Scene SIXCUBES. Time (x axis) vs. FFE (y axis). We compare classic method with new hierarchical method.*

4.3.3 Use of quasi-Monte Carlo sequences

Quasi-Monte Carlo sequences, described in section 2.2.8, are known to improve the performance of the Monte Carlo integration. These sequences have been tested in the Multipath algorithm (see section 3.3), producing reductions in the error to nearly the half. We have also tested some of these sequences (specifically Halton and Sobol sequences) in the context of the hierarchical algorithm presented in this chapter.

A simple scene containing a light source, a cube and a pyramid has been used in our tests. This scene is referred to as *Test1*. In Table 4.4 we present the execution time and the form factor error we have obtained. First column shows the number of lines used. Second column presents the execution time for the classical approach. The error for this classical approach (using Monte Carlo generation) appears on third column. Time for the new approach is on fourth column, whereas columns 5, 6 and 7 present the error obtained with the new approach using Monte Carlo, Halton and Sobol sequences, respectively. Note the reduction in error due to the use of quasi-Monte Carlo sequences. Relative efficiency for the new approach (using Monte Carlo generation and Halton and Sobol sequences) respect to the classical one is shown in Table 4.5. Note also that the speed-up factor is more than two times higher when using quasi-Monte Carlo sequences respect to ordinary Monte Carlo generation. Note also that Sobol sequences show a higher speed-up factor than Halton sequences. In Fig. 4.12 we present the graph of execution time versus form factor error for the classical method [47] (using Monte Carlo generation), the new approach introduced in this chapter using both Monte Carlo generation and Sobol sequences.

4.4 Conclusions

We have presented in this chapter a hierarchical approach to the Monte Carlo global method introduced in [47]. This new approach groups the objects in the whole scene in sub-scenes, obtaining a hierarchy of sub-scenes bounded by

LINES	T.CL.	FFE CL.	T.HI.	FFE HI.MC	FFE HI.H	FFE HI.S
$5 * 10^4$	12	0.331	22	0.0592	0.0346	0.0306
10^5	25	0.168	44	0.0315	0.0154	0.0119
$2 * 10^5$	50	0.084	88	0.0134	0.00686	0.00623
$4 * 10^5$	99	0.044	176	0.0074	0.00303	0.00270

Table 4.4: *Results for the scene TEST1. Second and fourth column show time for classic and hierarchical approach, respectively. Third column shows error for the classic approach. The 3 last columns show the error for the new approach using Monte Carlo, Halton and Sobol generation, respectively.*

LINES	REL.EFFIC.MC	REL.EFFIC.HALTON	REL.EFFIC.SOBOL
$5 * 10^4$	3.05	5.22	5.90
10^5	3.03	6.20	8.02
$2 * 10^5$	3.56	6.96	7.66
$4 * 10^5$	3.19	7.80	8.75

Table 4.5: *Relative efficiency (scene Test1) for the new approach using Monte Carlo generation and quasi-Monte Carlo Halton and Sobol sequences.*

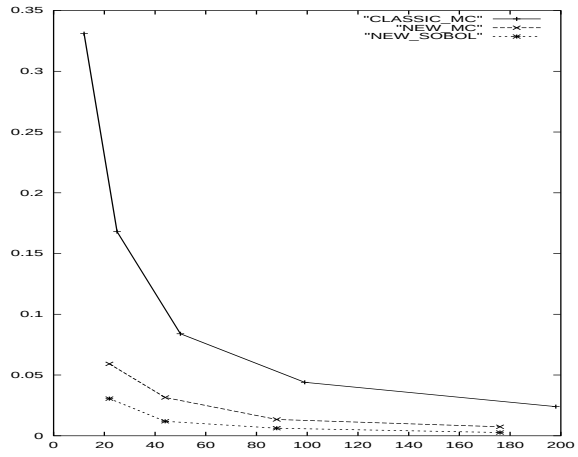


Figure 4.12: *Scene TEST1. Time (x axis) vs. FFE (y axis) for the classic method, the new approach using Monte Carlo generation and the new approach using Sobol sequences.*

spheres. This is used to generate densities of lines adapted to each sub-scene, allowing a better use of the lines. Form factors are estimated more accurately using these lines cast at the level of the sub-scenes (in addition to lines cast at the level of the whole scene). The increase in cost for these locally global lines is small in relation with the dramatic reduction of the error, resulting in a clearly better performance of the new approach, that presents a speed-up factor of about 3.

Furthermore, we have applied quasi-Monte Carlo sequences (Halton and Sobol) to the new algorithm. These sequences have produced an additional gain, improving clearly the results obtained with Monte Carlo generation. The speed-up factor has been multiplied by near 3 due to the use of the Sobol sequence. The performance of the Halton sequence is lower, but it multiplies the speed-up factor by more than 2.

Chapter 5

Hierarchical Transmittance-based Multipath

5.1 Introduction

The Multipath algorithm [51], described in 2.3.4, is a Monte Carlo technique that solves the radiosity problem, i.e. the illumination in a scene with diffuse surfaces. We introduce in this chapter a hierarchical variant of the Multipath algorithm using virtual bounding boxes. The communication between the different levels happens by these virtual boxes, storing on their transparent patches angular information. Several iterations must be done in order to distribute power stored in the virtual boxes.

This hierarchical approach reduces the cost of the Multipath algorithm by a factor of approximately 2. This reduction of the cost is obtained, on one hand, by accelerating the intersection cost for both local and global lines, and, on the other hand, by saving in memory and then reusing the random lines.

The results presented in this chapter basically correspond to [11].

5.2 Hierarchical approaches

The ideas of subdividing a complex scene in a hierarchy of sub-scenes and discretizing the directions over a surface have been widely used in rendering and particularly in radiosity. Let us summarize some of these approaches.

Some early radiosity approaches cited in [66] estimate the form factors from a patch by constructing an hemicube around the center of the patch. This hemicube is uniformly subdivided in a grid, and the rest of patches in the scene are projected onto the hemicube. The idea of subdividing the hemicube in a grid is similar to our subdivision of bounding boxes in virtual patches, but in these early approaches this is used to compute form factors and not to accumulate unshot power.

[19] introduces a parallel solution to the hierarchical radiosity method that allows to deal with very large environments. It divides the scene in single

groups of patches. During a single iteration power is bounced around between the patches within a group until convergence. No interaction with other groups occurs. After this internal balance, power is exchanged between other groups. This process can be repeated several times. This is somewhat similar to our algorithm in the sense that it alternates internal with external exchanges of power, but in [19] they use group iterative methods to solve the radiosity and form factors between clusters to interact at the external level.

[57] also uses a cluster hierarchy. Transmittance has been defined here as the ratio of power that passes through a cluster in a particular direction. That is, if the cluster is totally opaque in this direction, the transmittance is 0, whereas if some light can travel through the cluster in the direction, the transmittance is a positive value less or equal to 1. This is used to estimate the form factors in very complex environments consisting of a great number of small objects (vegetation environments). This approach considers the visibility function between two surfaces (patches) i and j to be constant. If the only occlusions between i and j are included in a cluster C, the form factor can be estimated as a product of the unoccluded form factor and the directional transmittance through C in the direction d_{ij} (d_{ij} being the mean direction between patches i and j). The same kind of complex scenes are treated in [56], where similar objects are replaced by instances of the same element. Thus this algorithm substitutes a very large hierarchical radiosity problem by a collection of smaller hierarchical radiosity problems.

Finally, [1] uses virtual walls in the context of parallel radiosity. Virtual walls are here used to separate different environments in which the main scene is subdivided.

5.3 Hierarchical Transmittance-based Multipath

We present here the *Hierarchical Transmittance-based Multipath* (HM) algorithm, a variant of the Multipath algorithm based on the subdivision of the scene in a hierarchy of sub-scenes. This allows running the Multipath algorithm not only for the whole scene but also for each sub-scene. Each sub-scene is bounded by a box subdivided in a structure of virtual patches and angular regions, that act as accumulators of incoming and outgoing power for the sub-scene. These accumulators are needed to connect the different levels. Before starting the exchange of power, a density of locally global lines is generated for each sub-scene (besides of the global lines generated for the whole scene). These lines allow to estimate the transmittance for each angular region. Transmittances give information about the opaqueness of the sub-scenes, permitting to ignore its interior when executing Multipath at the superior level, with the consequent reduction of cost. Additional reductions of the intersection cost for the local lines used in the first shot are also obtained because of the information obtained in this preprocess. Moreover, the intersection list for each line in the preprocess is saved in the RAM, allowing its later reuse. Next we detail each of the new algorithm features.

5.3.1 Hierarchy of sub-scenes

Several heuristics of subdivision can be applied to the scene. We have chosen here an algorithm based on a bottom-up strategy [21], although other algorithms could be used. We next describe this strategy.

We start by grouping from the set of single objects. Sub-scenes are created by grouping single objects and existing sub-scenes, being the new sub-scenes added to the set of non-grouped items. The grouping criterion is based on the quotient of the areas of the boxes. Namely, if the sum of the areas of the boxes that we want to group (note that single objects are also given a box) divided by the area of the resultant box is large enough (over a threshold), then we group in a new sub-scene. Note that this algorithm can produce any number of levels, depending on the geometry of the scene. Fig. 5.1 shows an example of a 3-level hierarchy. Note that each sub-scene is given a bounding box. Note also in the example that single objects do not always constitute a sub-scene: in other words, a sub-scene can contain other sub-scenes and/or single objects.

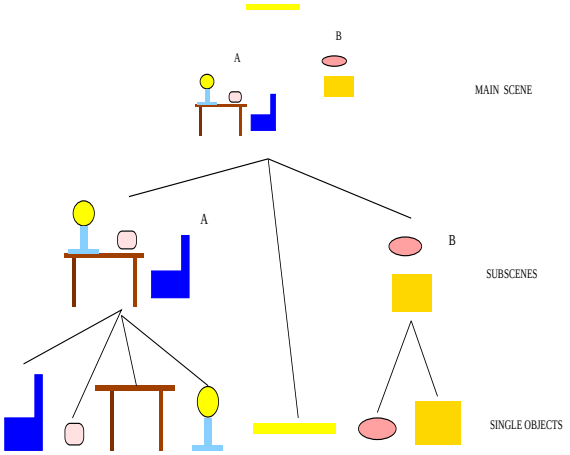


Figure 5.1: *A 3-level hierarchy of sub-scenes.*

5.3.2 Virtual bounding boxes

Each sub-scene is bounded by a virtual box (VBB). The 6 virtual walls of these VBB are subdivided in a grid of *virtual patches* (VP), as seen in Fig. 5.2 (left). Moreover, each hemisphere over a virtual patch is subdivided in *angular regions* (AR), so that the directions over the virtual patch are discretized (see Fig. 5.2 (right)). Each angular region will act as an accumulator of undistributed incoming and outgoing power. So, the angular regions participate in the balance of power between the inside and the outside of the box, connecting in this way the different levels of subdivision.

Note that the whole scene is usually bounded by real polygons (the walls) and thus, and since it constitutes the more external level, it does not need a virtual bounding box subdivided in VP and AR. This is its only difference with the sub-scenes. From here on, and for the sake of simplicity of the algorithm,

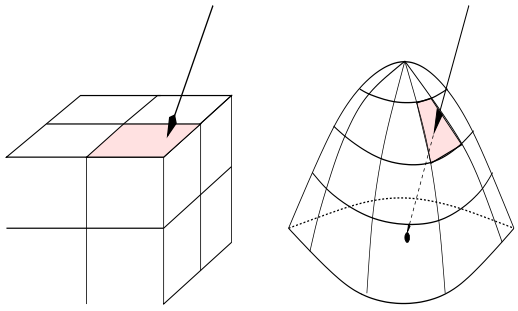


Figure 5.2: *(a) Subdivision of the bounding box faces in a grid of virtual patches. (b) Subdivision of each hemisphere over the center of a virtual patch in angular regions.*

we will consider the whole scene as an ordinary sub-scene.

We also have to note that, for a ray intersecting a bounding box, we can easily determine which virtual patch is intersected and, moreover, the corresponding angular region. Note that the angular region is determined just from the direction of the ray.

On the other hand, the more appropriate level of discretization in virtual patches and angular regions depends on each scene. In section 5.4.2 we describe the decisions we have taken in this sense. Note that some heuristic could be developed in the sense of establishing the most appropriate discretization, which is related to the number of lines to cast inside the sub-scene and, also, to the complexity of the sub-scene.

5.3.3 The preprocess

After building the hierarchy of sub-scenes and subdividing the bounding boxes into virtual patches and angular regions, it is time for a preprocess in which global lines are cast for each sub-scene. The information given by these lines is stored in the RAM, allowing their later reuse. For each line cast in sub-scene S, we have to compute its sorted-by-distance intersection list of angular regions and patches in the sub-scene. We consider the intersections of the line against:

- Single objects (non-grouped in further sub-scenes) inside S.
- Inner sub-scenes. In this case we ignore the inside of these sub-scenes, considering only their virtual walls and obtaining the corresponding angular regions. This strategy supposes an important reduction of cost, as described in section 5.4.
- Virtual walls of S, obtaining the angular regions. In case of S being the whole scene, we consider the real walls bounding the scene instead of virtual walls.

The obtained information about intersections is stored in the RAM. Note that in fact we store the numerical identifiers of intersected patches and angular regions (sorted by distance). Thus we have, for each line, an array of identifiers.

This information will be repeatedly used later, when running different levels of Multipath, so that we will avoid computing again the intersections.

The aim of this *preprocess* is to obtain information about the geometry of the sub-scenes. This information is used with two different purposes:

- **Estimation of the transmittances**.

  For each angular region R in the virtual box of a sub-scene S, its transmittance T_R (value between 0 and 1) represents the fraction of power entering S in the direction of R that goes through the sub-scene finding no obstacle in its way. Transmittances give an idea about the opaqueness of a sub-scene in each direction. The closer is T_R to 0, the more opaque is S in the directions associated to angular region R, and *vice versa*. Note that no transmittances are computed if S is the whole scene. Given an angular region R, its transmittance T_R is estimated as

$$T_R \approx \frac{n_R^d}{n_R} \tag{5.1}$$

  where n_R^d is the number of lines directly crossing R (hitting no object) and n_R is the total number of lines crossing R. Note that we use a density of lines specifically generated for the sub-scene, so that this density is uniform in the context of the sub-scene but not in the context of the whole scene. This is one of the main features of our algorithm. If no line crosses region R, we interpolate the value of the transmittances of the neighbouring regions. This allows us to obtain an approximation to the transmittances using a moderate number of lines.

  Transmittances will be used to avoid a loss of accuracy due to ignoring the interior of the sub-scenes. Note that transmittances give information about this interior (about its opaqueness in each direction). This strategy allows to accelerate the computation of the intersections list for each line. Transmittances could also be used to accelerate the distribution of primary power (first shot), avoiding to consider the interior of the sub-scenes, but we will rather distribute the primary power computing all the intersections to get a higher accuracy.

- **Obtaining the polygon associated to an angular region**.

  The second benefit obtained from lines cast in the preprocess allows to notably accelerate the first shot, that is, the expansion of primary power using local lines. It consists in computing, for each angular region in a bounding box, the polygon associated to this region. That is, if all lines cast in the preprocess intersect the same polygon p when crossing region R, we associate polygon p to region R. This means that a line entering the box by this region R is likely to have polygon p as the nearest intersected polygon, and then it makes unnecessary to test the line against the rest of the sub-scene. A variant of this strategy consists of considering not only a single associated polygon for region but several ones. This variant has been adopted in our implementation. Note that the maximum number of associated polygons is just an heuristic question.

Table 5.1: *Distribution of global lines, being N the number of local lines used in the first shot.*

Whole scene	65 per cent of N
Sub-scenes	25 per cent of N
Total global lines	90 per cent of N

5.3.4 Number of lines to cast in each sub-scene

The decision of how many lines to cast in each sub-scene (and also at the level of the whole scene) is just an heuristic question. Since the number of lines needed in the sub-scenes to obtain an accurate estimation of the transmittances is moderate, we have assigned low percentages of lines to the sub-scenes in relation to the percentage assigned to the whole scene. The percentage of lines for each sub-scene has been established from the complexity of the sub-scene, using the number of objects in the sub-scene as a naive measure of this complexity (although more sophisticated measures could be used [17]). Table 5.1 describes the exact percentages we propose from the number N of local lines used in the first shot. The percentage corresponding to the sub-scenes is distributed proportionally to the number of objects in each sub-scene.

5.3.5 Expanding the primary power: first shot

The distribution of primary power or *first shot* is done in the Multipath algorithm by using random local lines. These lines exit from light sources. Only nearest intersected polygon has to be computed. This computation can be accelerated by a factor of nearly 2 using the information obtained in the preprocess about polygons associated to regions. This means that if a local line gets into a sub-scene by angular region R, we only have to test the intersection of the line against the polygon (or polygons) associated to region R (if any). This allows to clearly reduce the cost of casting these local lines.

On the other hand, note that the primary power is not accumulated in the angular regions but in the patches. Angular regions will act as accumulators in the following stages, but not in the first shot, where it is preferred to exactly determine, for the sake of accuracy, the nearest intersected patch.

5.3.6 The iterative process

Once primary power has been expanded, it is time to estimate the indirect illumination. This stage corresponds to the execution of the Multipath algorithm at different levels. We need to accumulate incoming and outgoing power to/from the sub-scenes in the angular regions in which virtual bounding boxes are discretized. Note that this allows the communication between the different levels. Note that no new intersections have to be computed. Instead, we read from the RAM the intersections lists obtained in the preprocess. This implies that the cost of this stage is very low. Usually 4 or 5 iterations are enough to obtain an acceptable result.

5.3.7 The Hierarchical Multipath algorithm

The hierarchical algorithm is presented in Fig. 5.3. Note that MP recursive function deals with the exchange of power inside and between each sub-scene in the hierarchy. In Fig. 5.4 we present the algorithm corresponding to this recursive function (the abbreviations VP, AR, and VBB correspond to virtual patches, angular regions and virtual bounding boxes, respectively).

Generate hierarchy of sub-scenes
Subdivide each virtual box in VP and AR
for each *sub-scene S (including the whole scene)*
  Cast global lines and store intersection lists
  if *S is not the whole scene*
    Compute transmittances and associated polygons for each AR in S
  end if
end for
First shot (using polygons associated to each AR to accelerate it)
for each *iteration (4 or 5 are usually enough)*
    MP(whole scene) // recursive function
end for

Figure 5.3: *The HM algorithm.*

Referent to the HM algorithm, we have to remark the following points:

- Since we have cast lines in a preprocess, we know in advance the number of lines that cross each patch and angular region (used to compute power per line).
- The "if any" nuance refers to the fact that the whole scene has not necessarily got virtual bounding boxes.
- The computation of primary power per line has sense only at first iteration.
- Transmittances are used to establish the fraction of power crossing the sub-scenes in each direction, allowing to ignore the interior of the sub-scenes without losing much accuracy.

5.4 Results

5.4.1 Tested scenes

We have tested two different scenes, that will be described next:

- **Scene** *ROOM*. This scene represents a room with a table, some chairs and a desk, and several small objects on the table and on the desk. Scene *ROOM* has been discretized in about 11000 patches.
- **Scene** *OFFICE*. This scene represents a room with several desks, chairs and shelves. It has been discretized in approximately 7000 patches.

```
function MP(scene S)
for each patch i in S
   if i not included in any sub-scene of S
      Compute primary power per line for i
   end if
end for

if S is not the main scene
   for each AR in the VBB of S
      Compute incoming power per line
      Set to 0 outgoing accumulated power
   end for
end if

for each sub-scene s inside S (only at next level in the hierarchy)
   for each AR in the VBB of s
      Compute outgoing power per line
      Set to 0 incoming accumulated power
   end for
end for

for each line cast in S
   Read intersection list L
   for each item e in L
      case
         e is a patch: Send its unshot power + its power per line
         e is an AR of S: Send its incoming power per line
         e is an AR of a sub-scene of S: Send its outgoing power per line
      end case
   end for
   Power leaving S is accumulated as outgoing power in the corresponding AR
   Power crossing a sub-scene of S is accumulated as incoming
         power in the corresponding AR and attenuated by the
         corresponding transmittance
end for

for each sub-scene s inside S (only at next level in the hierarchy)
   MP(s)                  // recursive call
end for
end function
```

Figure 5.4: *The recursive MP function.*

Table 5.2: *Percentages of global lines used in scene ROOM (from number of local lines used in the first shot)*

External Multipath	65 per cent
Sub-scene B (table, chairs, and objects on table)	16.6 per cent
Sub-scene C (desk, and objects on desk)	8.4 per cent

The geometry of scene *ROOM* marks clearly two sub-scenes: one of them includes the desk and the small objects of its environment, and the other includes the table and chairs with their corresponding small objects. Light source is considered as a single object. Five sub-scenes have been established in scene *OFFICE*. Two of them correspond to both shelves, and the other three group desks and chairs. Light source is also considered as a single object.

5.4.2 Virtual patches and angular regions

Each face in the virtual boxes bounding the sub-scenes has to be subdivided in a grid of virtual patches, which at the same time have to be subdivided in angular regions. Different tests have been carried out to establish the best level of subdivision in our scenes.

For our test scenes, the optimal results have been obtained by subdividing each face of a virtual box in a grid of 4×4 virtual patches, and the hemisphere over each virtual patch in 8 angular regions (longitude is divided in 4 parts and latitude in 2 parts).

High levels of subdivision (in virtual patches and AR) could be appropriated when dealing with very complex environments, but in our scenes it appears to be unnecessary, since it strongly increases the number of lines needed to estimate transmittances with an acceptable accuracy.

5.4.3 Number of lines

We have used the heuristic criterion described in 5.3.4, that fixes the percentage of lines at the top level and assigns to each sub-scene a number of lines proportional to the number of objects that contains. Exact percentages for both scenes *ROOM* and *OFFICE* are described in Tables 5.2 and 5.3, respectively. Note that in both cases the percentages refer to the number of local lines used in the first shot.

Note that the total number of global lines used for both scenes is a 90 per cent of the number of local lines, whereas in classical Multipath we use the same number of local and global lines. This reduction in the number of global lines cast is due to the adaptation of the densities of lines to each sub-scene.

5.4.4 Computing the Mean Square Error

Mean square error (MSE) has been used in our comparisons. We have compared the obtained radiosities with the reference values. These reference values have been obtained using a very high number of lines. We have to remark here that there exists a bias between the results obtained with classic Multipath

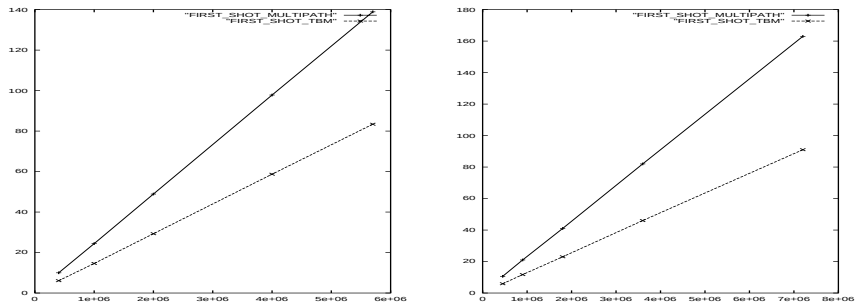
Table 5.3: *Percentages of global lines used in scene OFFICE (from number of local lines used in the first shot)*

External Multipath	65 per cent
Sub-scene B (3 desks and chairs)	10.8 per cent
Sub-scene C (2 desks and chairs)	7.2 per cent
Sub-scene D (1 desk and chair)	3.6 per cent
Sub-scene E (shelf)	1.8 per cent
Sub-scene F (shelf)	1.8 per cent

and the ones obtained with HM. This bias, produced by the discretization in virtual patches and AR, does not bear any noticeable visual difference, so it is considered acceptable. To avoid the bias effects in the comparisons, we have computed the error with respect to the reference values generated using each algorithm (HM and classic Multipath).

5.4.5 Reduction of the first shot cost

The information obtained from the preprocess about polygons associated to the angular regions allows to reduce the cost of the local lines involved in first shot in our HM. Note that this information is obtained nearly for free, since lines cast in the preprocess are stored in the RAM and reused later. In Fig. 5.5 we present graphs of number of local lines used in first shot versus first shot time for both scenes *ROOM* (left) and *OFFICE* (right). In these graphs we can observe the noticeable reduction of first shot time using HM. Particularly, first shot time has been reduced by a 40 per cent in scene *ROOM*. In scene *OFFICE*, first shot time has been reduced by a 44 per cent.



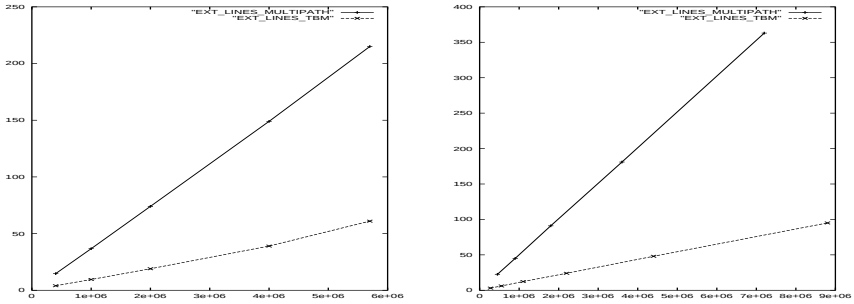
(a) (b)

Figure 5.5: *First shot. Graphs of number of local lines (horizontal axis) vs. time (vertical axis) (a) Scene ROOM. (b) Scene OFFICE.*

5.4.6 Reduction of cost due to skipping the interior of sub-scenes

As mentioned in 5.3.3, and thanks to the use of angular regions as accumulators of power and also to the use of the transmittances, HM algorithm allows to skip the interior of the sub-scenes when casting lines at the superior level. This bears a noticeable reduction in the cost of casting global lines, since it eliminates part of the costly intersections.

In our scenes, this affects to lines cast at the top level, that is, at the level of the whole scene. Fig. 5.6 presents graphs of number of lines versus time for both scenes *ROOM* (left) and *OFFICE* (right). In these graphs we observe the clear reduction of cost obtained in lines cast at the top level when using HM. Particularly, execution time has been reduced by approx. a 75 per cent in scene *ROOM* and by approx. a 78 per cent in scene *OFFICE*.



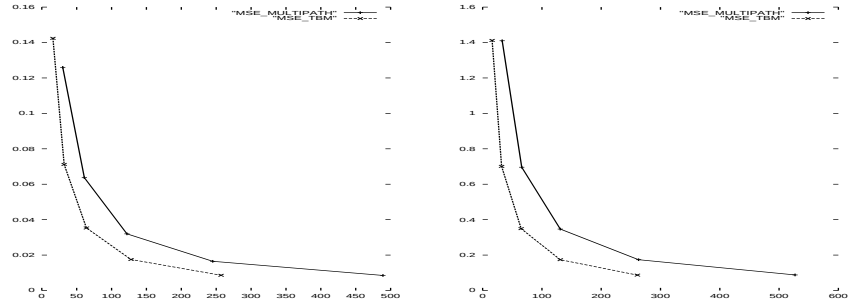
(a) (b)

Figure 5.6: *Lines at top level. Graphs of number of lines (horizontal axis) vs. time (vertical axis) (a) Scene ROOM. (b) Scene OFFICE.*

5.4.7 Summarizing the gain of HM

The new HM algorithm obtains noticeable reductions of first shot local lines and global lines costs. Moreover, the total number of global lines required in HM is lower than the one in classic Multipath. Another feature of the new algorithm is the storage in the RAM and later reuse of the intersections lists. All these factors contribute to the better performance of HM in front of classic Multipath.

The new HM algorithm obtains a speed-up factor of approximately 2 in front of classic Multipath, as it can be noticed from the results in Tables 5.4 and 5.5, and in graph in Fig. 5.7. For the scene *ROOM* we present in Fig. 5.7 (left) the graph of time vs. MSE. Fig. 5.10 allows to compare the image obtained with classic Multipath (a) and the obtained with HM (b). Their respective running times and number of lines for each stage appear on Table 5.4. For scene *OFFICE* we also present the graph of time vs. MSE (fig. 5.7 (right)) and images obtained with both methods (Fig. 5.11 (a) and (b)). Running times and number of lines for these images appear on Table 5.5.



(a) (b)

Figure 5.7: *Graphs of total running time (horizontal axis) vs. MSE (vertical axis) for (a) Scene ROOM. (b) Scene OFFICE.*

Table 5.4: *Scene ROOM. Execution times and number of lines used to generate the images in Fig. 5.10. Note that the speed-up factor is* 1.9 *in the case of using Monte Carlo generation and* 3.2 *in the case of using quasi-Monte Carlo generation*

	CL. MULTIPATH	HM (MC)	HM (HALTON)
First shot	8 M lines	8 M lines	4.7 M lines
First shot time	195 sec.	117 sec.	70 sec.
Global lines	8 M lines	7.2 M lines	4.3 M lines
Global lines time	294 sec.	116 sec.	70 sec.
Reusing lines time		24 sec.	14 sec.
TOTAL LINES	16 M lines	15.2 M lines	9 M lines
TOTAL TIME	489 sec.	257 sec.	154 sec.

Table 5.5: *Scene OFFICE. Execution times and number of lines used to generate the images in Fig. 5.11. We obtain a speed-up factor* 2.0 *when using Monte Carlo generation and* 3.5 *when using quasi-Monte Carlo generation.*

	CL. MULTIPATH	HM (MC)	HM (WEYL)
First shot	3.6 M lines	3.6 M lines	2.1 M
First shot time	82 sec.	46 sec.	26 sec.
Global lines	3.6 M lines	3.25 M lines	1.9 M
Global lines time	181 sec.	72 sec.	42 sec.
Reusing lines time		13 sec.	8 sec.
TOTAL LINES	7.2 M lines	6.85 M lines	4 M
TOTAL TIME	263 sec.	131 sec.	76 sec.

5.4.8 Analysis of the error

The previous results manifest the superiority of the new HM algorithm in front of classical Multipath. A speed-up factor of about 2 has been obtained. Although the new method carries some bias due to the reuse of the lines and the angular discretization, this bias practically has no visible effect on the final image. However, we have studied and represented in Fig. 5.8 the relative error distribution for the scene *ROOM*. White patches represent the zero error in each color channel. Black patches have an error about 2 per cent for all the color channels. Note that the maximum error occurs near the virtual boxes. These artifacts are due to the discretization in virtual patches and angular regions.

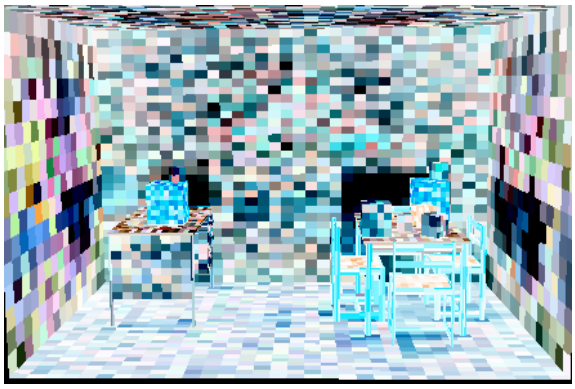


Figure 5.8: *Scene ROOM. Distribution of the relative error. Darker zones present a higher relative error respect to the converged image.*

5.4.9 Use of Quasi-Monte Carlo sequences

We have incorporated quasi-Monte Carlo generation to the HM algorithm. As presented in chapter 3, the use of quasi-Monte Carlo sequences improves significantly the performance of the Multipath algorithm. Since the improvement by the use of quasi-Monte Carlo is independent to the one introduced in this chapter, an additional gain is to be expected when adding quasi-Monte Carlo sequences.

In our test we have obtained the best results using Halton and Weyl sequences. The error has been reduced to nearly the half with respect to the use of classical Monte Carlo generation. In Tables 5.6 and 5.7 we present the reduction of the MSE due just to the use of quasi-Monte Carlo sequences. Note that the reduction of the MSE is similar to the one produced when using quasi-Monte Carlo in the context of classic Multipath algorithm.

Both HM algorithm and quasi-Monte Carlo generation have produced a speed-up factor between 3 and 4 when applied together. We present in Fig. 5.9 the graph time vs. MSE for both scenes *Room* and *Office*. Resulting images are presented in Fig. 5.10 (for scene *Room*) and in Fig. 5.11 (for scene *Office*). Note that combining both HM and quasi-Monte Carlo we obtain the same MSE with an important reduction of cost, due to both the reduction of number of lines needed to get the same accuracy and the reduction of cost for local and

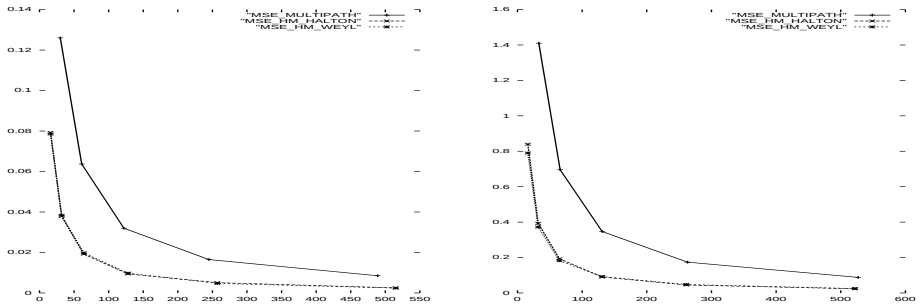
Table 5.6: *Scene ROOM. Comparison of the MSE due to the use of quasi-Monte Carlo sequences vs. Monte Carlo in the HM algorithm. Note that, for the same number of lines -and practically the same cost- the MSE obtained using Halton and Weyl sequences is reduced to nearly the half.*

NUMBER OF LINES	MONTE CARLO	HALTON	WEYL
0.95 M	0.1424	0.0793	0.0785
1.9 M	0.0712	0.0378	0.0384
3.8 M	0.0353	0.0192	0.0198
7.6 M	0.0175	0.0094	0.0098
15.2 M	0.0086	0.0051	0.0047

Table 5.7: *Scene OFFICE. Comparison of the MSE due to the use of quasi-Monte Carlo sequences vs. Monte Carlo in the HM algorithm. Note that, for the same number of lines -and practically the same cost- the MSE obtained using Halton and Weyl sequences is reduced to nearly the half.*

NUMBER OF LINES	MONTE CARLO	HALTON	WEYL
0.86 M	1.412	0.841	0.792
1.71 M	0.701	0.393	0.372
3.43 M	0.349	0.193	0.182
6.85 M	0.174	0.090	0.093
13.7 M	0.087	0.045	0.048

global lines. Number of lines used and time for each stage are described in Tables 5.4 and 5.5.



(a) (b)

Figure 5.9: *Use of HM algorithm together with qMC sequences. Graphs of total running time (horizontal axis) vs. MSE (vertical axis) for (a) Scene ROOM. (b) Scene OFFICE. In the graphs are compared classic Multipath versus HM with quasi-Monte Carlo generation.*

5.4.10 Technical specifications

All the executions have been done on a Pentium-IV 1.6 Ghz with 1 Gb of RAM. The RAM requirements to store the intersections lists have to be considered. Each line at top level requires approximately 20 bytes, whereas lines at the sub-scenes require approximately 40 bytes. That means that the storage of intersections lists in the execution corresponding to image presented in Fig. 5.10 (right), where we have used 6 millions of external lines and 1.2 millions of internal lines, has required approximately 170 Mb of RAM.

5.5 Conclusions

We have presented a new algorithm based on the global line Multipath method [51]. This algorithm uses a bounding boxes hierarchy, establishing sub-scenes in which Multipath is locally executed. Thus we distinguish Multipath executions at different levels. Virtual boxes bounding the sub-scenes are subdivided in virtual patches and angular regions, that act as accumulators of incoming and outgoing power, linking the different levels. The sub-scenes hierarchy allow to adapt the density of lines, resulting in a reduction of the total number of global lines needed. On the other hand, transmittance is calculated for each angular region. Transmittances give an estimation of the transparency of boxes in each direction. It allows to accelerate global lines without noticeable loss of accuracy. Moreover, information about polygons associated to each region allows to accelerate the first shot. Finally, saving in memory the intersections lists allows to reuse the lines, thus saving in cost.

These features of the HM algorithm produce a speed-up factor of approximately 2 respect to classic Multipath in the scenes we have tested. This better performance obeys to the reduction of cost in local and global lines, to the reduction of the total number of global lines needed and to the reuse of the global

lines. Moreover, when adding quasi-Monte Carlo generation, the speed-up factor increases to about 3.5.

We have to remark the similarities of this algorithm with respect to the one presented in chapter 4. Both algorithms are based on subdividing the environment in a hierarchy of sub-scenes and adapting the densities of random global lines to each sub-scene. Thus both algorithms use *locally global lines*. However notable differences appear between both algorithms. The main difference -apart of the fact that one estimates form factors and the other simulates the distribution of power- refers to the ambit where the locally global lines are valid. In the case of the HM algorithm, lines are valid just inside the sub-scene where they are cast. Instead, in the algorithm presented in chapter 4, lines cast in a sub-scene are valid to estimate form factors from patches in the sub-scene to patches out of the sub-scene. Thus, this algorithm requires considering not only the interior of the sub-scene where lines are cast but also the outside. This makes the HM algorithm easier to be parallelized.

Figure 5.10: *Scene ROOM with MSE= 0.0095. (top) Classic Multipath. Running time: 489 sec. (center) Hierarchical Multipath. Running time: 257 sec. Speed-up factor is nearly 2. (bottom) Hierarchical Multipath using Halton sequences. Running time: 154 sec. Speed-up factor is over 3.*

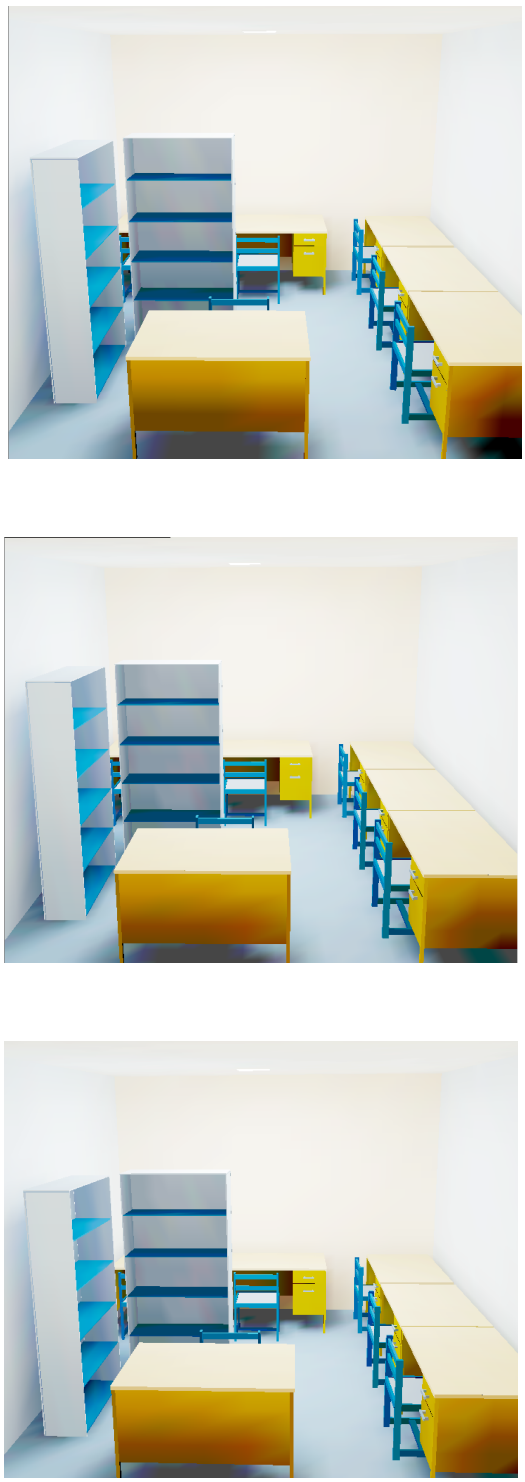


Figure 5.11: *Scene OFFICE with MSE= 0.174. (top) Classic Multipath. Running time: 263 sec. (center) Hierarchical Multipath. Running time: 131 sec. Speed-up factor is 2. (bottom) Hierarchical Multipath using Weyl sequences. Running time: 76 sec. Speed-up factor is about 3.5.*

Chapter 6

Extended Ambient Term

6.1 Introduction

This chapter presents a new approach to the idea of ambient term, the *extended ambient term*. Our motivation is to obtain the maximum information from the scene without the computation of occlusion conditions. The classic ambient term operator will be substituted by another more sophisticated operator that splits the ambient computation into a small number of classes established according to orientation criteria. After the classification of the surfaces in the scene, a small linear equations system deals with the power exchange among the classes, computing an ambient term for each class. This means that the rendering problem is highly simplified by computing the inter-reflections between only a reduced number of classes, instead of between all the elements in the scene. This approach is slightly more complicated than the classic ambient term, but it allows to obtain a noticeable gain in the solution with a negligible increase in computational cost. Although we only present the application to diffuse environments, the method has a trivial generalization to non-diffuse case.

The extended ambient term can be applied to global illumination methods that compute the rendering equation expansion in Neumann series -in an explicit or implicit way- to fast distribute the undistributed power. We will apply here the extended ambient term to the radiosity context.

The results of this chapter have been published in [7].

6.2 Ambient term

Ambient term has been widely used in computer graphics to visualize the non-computed higher order reflections, improving the appearance of the image. It is used in most simple graphics software packages to show the non-directly lighted and thus invisible shadow parts. In radiosity and global illumination literature it is used to visualize the effect of the undistributed or residual power. Ambient term ignores the geometrical occlusion conditions, resulting in a negligible computational cost.

Ambient term was first used in computer graphics in first shadowing models [43]. The user could interactively give the intensity of the ambient term to visualize the parts in total shadow. In [12] it was introduced into global

illumination computations. As seen in chapter 2, global illumination model is described by the rendering equation (see equation 2.1). This equation can be formally inverted using Neumann series

$$L = (I + \mathcal{R} + \mathcal{R}^2 + \mathcal{R}^3 + ...)L_e = L_e + \mathcal{R}L_e + \mathcal{R}^2 L_e + ... + \mathcal{R}^k L_e + ... \quad (6.1)$$

where the k-th order term in the expansion represents the exiting power after k reflections.

The idea of (classic) ambient term is to avoid totally or partially the complicated visibility computations by substituting the expansion in (6.1) beyond a given term by a constant operator (more precisely, by the identity operator times a constant, the area weighted average reflectance). This single constant has an effect on all the scene, being called the *ambient term*. With this strategy, the distribution of the remaining undistributed power is done almost for free.

6.3 Classic ambient term applied to the radiosity method

Before introducing the new extended ambient term in the radiosity context, let us remind the formalization of the classic ambient term when applied to the radiosity context. The radiosity system of equations (2.4) can be written [55] in matrix form as

$$B = E + \mathcal{R}B \quad (6.2)$$

where B is the vector of radiosities, E is the vector of emittances (self-illumination) and the operator $\mathcal{R}$ -that incorporates the form factors and the reflectances- solves for direct and indirect illumination. Note that the above equation corresponds to the rendering equation (2.1) in the particular case of radiosity, and so it allows a representation in Neumann series

$$B = (I + \mathcal{R} + \mathcal{R}^2 + \mathcal{R}^3 + ...)E = E + \mathcal{R}E + \mathcal{R}^2 E + ... + \mathcal{R}^k E + ... \quad (6.3)$$

where

- The zero-order term E corresponds to self-illumination.
- The first-order term $\mathcal{R}E$ corresponds to direct illumination.
- The rest of the terms correspond to indirect illumination (second, third, etc., order). The order of operator $\mathcal{R}$ in the expansion indicates the number of reflections experimented by the light.

The radiosity system of equations can be solved by using several numerical methods that intend to obtain the distribution of luminous power by simulating the successive inter-reflections. These methods perform this simulation until a given step, so that an undistributed power always remains. The idea of ambient term approximation [12, 39] is to make an immediate distribution of this undistributed power by computing a single quantity, the ambient radiosity, that

represents the incoming radiosity due to the ambient. In this way, the expansion in equation (6.3) is substituted from a determinated term by a constant operator, in which the form factors are not considered, the reflectance of every patch is substituted by the average reflectance of the scene, and the undistributed power is accumulated in a total. Let

- $A^{total} = \sum_i A_i$ be the total area of the surfaces in the scene.
- $\bar{\rho} = \frac{\sum_i A_i \rho_i}{A^{total}}$ be the average reflectance.
- $U^{total} = \sum_i U_i$ be the total undistributed power.
- $B^{AMB(in)}$ be the incoming ambient radiosity.

Then we have

$$B^{AMB(in)} = \frac{U^{total}}{A^{total}(1-\bar{\rho})} \tag{6.4}$$

Note that the total undistributed power is expanded by the term $\frac{1}{(1-\bar{\rho})}$ that corresponds to the series $1 + \bar{\rho} + \bar{\rho}^2 + ...$ obtaining the incoming ambient power. This power is divided by the total area, and the amount obtained is the incoming ambient radiosity of all the scene $B^{AMB(in)}$.

From this incoming ambient radiosity, the outgoing ambient radiosity corresponding to each patch i is computed by multiplying by the reflectance of the patch

$$B_i^{AMB(out)} = \rho_i B^{AMB(in)} \tag{6.5}$$

and finally the total outgoing radiosity of each patch is the sum of this ambient radiosity and its outgoing accumulated radiosity B_i^{accum} (this amount is the accumulated radiosity before adding the ambient term and including the emittance)

$$B_i = B_i^{accum} + B_i^{AMB(out)} \tag{6.6}$$

Ambient term was initially thought to be applied at the last stage of the simulation as a final correction or make-up to the obtained image. But it can also be applied just at the beginning of the radiosity process as a very fast first approximation to the solution; then the power of the light sources constitutes the undistributed power.

A new approach to the classic ambient term was suggested in [36]. This consists in grouping the surfaces in the scene according to their normal vectors. We will next develop this approach.

6.4 Extended ambient term

The new extended ambient term, like the classic ambient term, will be applied to the radiosity method to fast distribute the undistributed power and so to obtain a smoother final image. The main idea involved in it is the substitution of a

constant operator, the classic ambient term seen in 6.3, by a more sophisticated new operator, the *extended ambient term*. This new operator, unlike the classic one, incorporates slight geometrical considerations (but not occlusions). The polygons in the scene are classified into a small number C of classes according to their position (that is, according to their normal vectors). The distribution of the undistributed power of the classes will be fast estimated by solving the radiosity system of equations among these classes. The unknowns of this system will be the ambient terms for each class.

In this way, the classic ambient term reviewed in 6.3 will be improved in the sense of considering more information. Instead of having only one average reflectance for the whole scene, we will work with one average reflectance for each class, and instead of considering one undistributed power for the whole scene, one undistributed power for each class will be used. We have to consider the power interaction between the C classes to obtain the ambient term for each class.

6.4.1 A first approach

Every polygon (and so every patch in the polygon) has to belong to one of the C classes. Each class is represented by a normal vector that indicates its position. The membership of a polygon to a class is given by the normal vector of the polygon: the polygon belongs to the class that has the most similar normal vector (corresponding to the greatest dot product between normal vector of the polygon and normal vector of each class).

Once every polygon has been assigned to a class, we must compute for each class the undistributed power, the average reflectance and the total area. Let

- $A_k^{cl} = \sum_{i \epsilon k} A_i$ be the total area of class k.
- $\bar{\rho}_k = \frac{\sum_{i \epsilon k} A_i \rho_i}{A_k^{cl}}$ be the average reflectance of class k.
- $U_k^{cl} = \sum_{i \epsilon k} U_i$ be the total undistributed power of class k.

Then, for every class k, we establish the radiosity equation, where the unknown B_k^{total} is the total outgoing radiosity corresponding to class k. We have to consider the form factors F_{kj} between the classes, that will be described in 6.4.2. The radiosity equation is then:

$$B_k^{total} = \frac{U_k^{cl}}{A_k^{cl}} + \bar{\rho}_k \sum_{j \epsilon 1..C} B_j^{total} F_{kj} \tag{6.7}$$

That leads to a linear system of C equations. Solving this system, we obtain, for each class k, the total outgoing radiosity B_k^{total}. From here, we compute the outgoing ambient radiosity of the class $B_k^{AMB(out)}$ by subtracting the undistributed radiosity:

$$B_k^{AMB(out)} = B_k^{total} - \frac{U_k^{cl}}{A_k^{cl}} \tag{6.8}$$

Dividing this outgoing ambient radiosity by the average reflectance of the class we obtain the incoming ambient radiosity for each class. This ambient radiosity is valid for all the patches in the class.

$$B_k^{AMB(in)} = \frac{B_k^{AMB(out)}}{\bar{\rho}_k} \tag{6.9}$$

The outgoing ambient radiosity of patch i will be obtained just multiplying the incoming ambient radiosity of the class by the reflectance of patch i

$$B_i^{AMB(out)} = \rho_i B_i^{AMB(in)} \tag{6.10}$$

Finally, this ambient radiosity is incorporated to the outgoing radiosity of the patch.

6.4.2 Form factors between the classes

To simplify both the estimation of the form factors and the solving of the system, the number of classes has been fixed to 6, with each class identified with the face of a cube. Thus, in analogy with the faces of a cube, we set the value of the form factor between two different classes to 0.2 and between a class and itself to zero (these are the almost exact values of form factors between the faces of a cube).

Note that this supposes an analogy with a six-sided environment similar in color and area to the real scene, in which we compute the inter-reflections. The proposed values for the form factors are just a coarse approximation, but it is enough to obtain acceptable results. However, a different number of classes could be used.

6.4.3 A more sophisticated approach: use of fuzzy classes

Using the first approach, every patch belongs to one and only one class. This link is quite strong, in the sense that it can produce a loss of geometrical information about the patches. Consider Fig. 6.1. According to the approach in the left side, patch i belongs only to class 2. But it is clear that this patch also has a relation with class 1 that in the first approach is missed.

The solution to this problem is presented in this second approach: the use of *fuzzy classes*. Now, the patches are not linked to a single class. They are related to several classes by means of weights. That is, every patch i has a weight in each class k, μ_{ik}. The greater is the relation of patch i to class k, the greater will be the weight (if there is no relation, the weight will be 0). The sum of the weights will be 1 for each patch i:

$$\sum_{k\epsilon 1..C} \mu_{ik} = 1 \tag{6.11}$$

We can see an example on the right part of Fig. 6.1. Note that now we use more geometric information than in the first approach. The use of these fuzzy classes produces more accurate results with a negligible increase in cost, as we will see in section 6.5.

We must establish the weights for each patch in the scene. We use the square cosine of the angle between the normal of the patch and the normal of the class, in the case of positive cosine (if the cosine is negative, the weight is set to zero). Since we use 6 axis-aligned classes, the sum of the weights for each patch will be 1. Note that each patch belongs to a maximum of 3 classes.

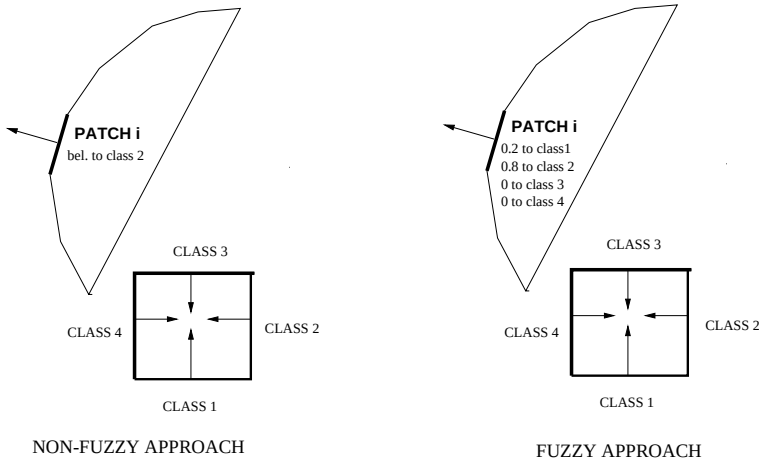


Figure 6.1: *Left: non-fuzzy classes: patch i only belongs to class 2. Right: fuzzy classes: patch i has a weight for each class.*

Thus, the main difference between this second approach and the first one is that in the former one there was a 0-1 relation (belonging or not belonging) between patches and classes, whereas now this relation is established by weights between 0 and 1. This fact has to be reflected in the formulation. Let

- $A_k^{cl} = \sum_i \mu_{ik} A_i$ be the total area of class k.
- $\bar{\rho}_k = \frac{\sum_i \mu_{ik} A_i \rho_i}{A_k^{cl}}$ be the average reflectance of class k.
- $U_k^{cl} = \sum_i \mu_{ik} U_i$ be the total undistributed power of class k.

The radiosity system of equations between the classes will be established in the same way as in the non-fuzzy case (6.7). This leads to the estimation, by solving the system, of the total outgoing radiosity and, subsequently, of the outgoing ambient radiosity for each class, as seen in equation (6.8). From here we can compute the incoming ambient radiosity for each class (6.9). Unlike the first approach, where the incoming ambient radiosity for every patch matched with the one of the class which the patch belonged to, in this fuzzy approach we must compute this amount as a weighted average of the ambient terms of all the classes:

$$B_i^{AMB(in)} = \sum_{k \epsilon 1..C} \mu_{ik} B_k^{AMB(in)} \tag{6.12}$$

Finally we obtain the outgoing ambient radiosity for each patch multiplying by its reflectance. Note that we can consider the first approach in 6.4.1 as a particular case of the fuzzy approach.

6.4.4 Using a hierarchy of sub-scenes

The use of a hierarchy of sub-scenes bounded by virtual boxes, presented in chapter 5 in the context of the Multipath algorithm, can be incorporated to the

extended ambient term. The idea is to manage a set of 6 classes -matching with the virtual walls of the bounding boxes-, with their respective ambient terms, for each sub-scene, instead of using a single set of classes for the whole scene. In this way, we have to solve the extended ambient term system of equations for each sub-scene. This process can be iterated, considering in each iteration the corresponding undistributed power for each sub-scene (and also for the whole scene). The increase in cost due to the use of the hierarchy of boxes is nearly negligible. The use of the hierarchy of sub-scenes allows to deal with the ambient radiosity at the level of each sub-scene, resulting in more significative images, as seen in 6.5. Next we present the algorithm (Fig. 6.2).

```
Generate hierarchy of sub-scenes
for each iteration (3 or 4 are usually enough)
      extendedAmbientTerm(whole scene)      // recursive function
end for
```

Figure 6.2: Extended ambient term with hierarchy of sub-scenes.

Note that in each iteration we call the recursive function *extendedAmbient-Term*, that solves for the extended ambient term at all levels of the hierarchy. Let us present this function (Fig. 6.3)

```
function extendedAmbientTerm(scene S)
   Compute undistributed power for each of the 6 classes of scene S
   Solve radiosity system of equations for the classes of scene S
   Add the ambient terms to the corresponding patches and virtual walls
   for each sub-scene B inside S (only at next level in the hierarchy)
       extendedAmbientTerm(B)      // recursive call
   end for
end function
```

Figure 6.3: The recursive extendedAmbientTerm function.

The undistributed power for each class of S arises from the contribution of the virtual walls of S (incoming power) and of the virtual walls of the sub-scenes of S (outgoing power). Only in the case of the first iteration we have to consider the emitted power from the light sources in S not included in further sub-scenes.

On the other hand, we must not only add the computed ambient terms to the patches in S not included in further sub-scenes, but also to the virtual walls of S as outgoing power and to the virtual walls of the sub-scenes of S as incoming power.

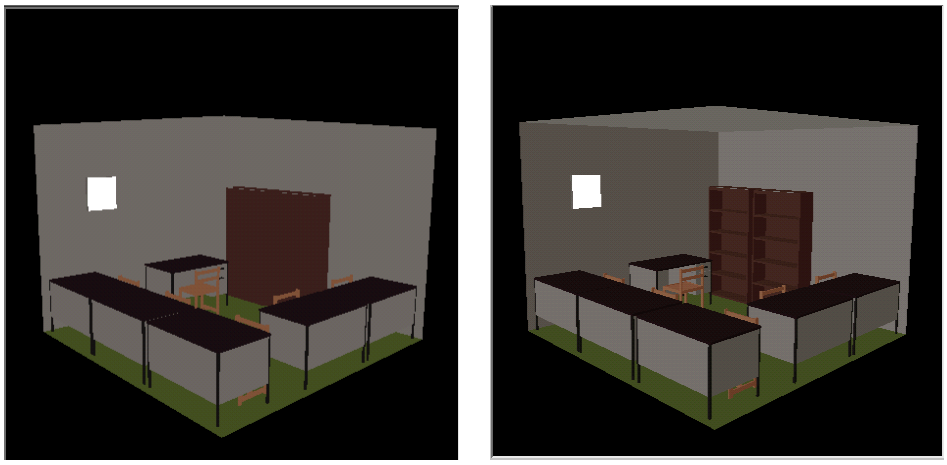
Note that the 6×6 linear system of equations, corresponding to the balance of power between the classes, has to be solved, in each iteration, for each sub-scene. The computation of undistributed power, mean reflectance and area for each class is done in the same way as described above.

Note also that, since we do not consider directionality when dealing with the extended ambient term, it is not necessary the subdivision of the virtual walls of

the boxes in virtual patches and angular regions, as done in chapter 5. We only accumulate incoming and outgoing power in the virtual walls. Transmittances are not necessary either.

6.5 Results

The application of the extended ambient term introduces a great improvement in the ambient illumination in front of the classic ambient term. The use of C ambient terms, being C the number of classes (6 in our case), instead of a single term gives a greater richness to the obtained images. This fact can be observed in Fig. 6.4. The image obtained with the extended ambient term (on the right) offers us more information than the one on the left, due to the contribution from each class. Note that in this example no direct illumination has been computed, so the radiosity for every patch is only due to the ambient term and, in the case of light sources, to the emittance. The increase in execution time in the extended ambient term case is very small, from 0.24 to 0.29 seconds. All images in this chapter were computed on a Pentium II at 350 Mhz.



(a) (b)

Figure 6.4: *(a) Classic ambient term. Execution time: 0.24 sec. (b) Extended ambient term. Execution time: 0.29 sec.*

6.5.1 Color bleeding

The extended ambient term provides color bleeding effects. They can be observed in Fig. 6.5. In this figure we have a grey cubic room with a red wall and an opposite green wall. In the center of the room there is a white cube, and the scene is illuminated by a square lamp stuck on the ceiling. We can observe the color bleeding in the faces of the cube in front of the colored walls. This image has been computed taken into account direct illumination. Note that the extended ambient term accounts here for indirect illumination.

These color bleeding effects are due to the null contribution of a class to itself in the radiosity system of equations (remind that the form factor $F_{kk} = 0$). Thus, in Fig. 6.5, the reddish face of the cube belongs to the same class as the green wall, and so it is influenced by the other 5 classes, but not by the own. This explain the reddish color (the same occurs with the greenish face).

Note that the color bleeding effects are related with the reflectances of the classes. For instance in the scene of Fig. 6.5, where the contribution of each wall to the reflectance of the classes is very important, changing the color of the walls could cause the color bleeding to disappear or even be inverted.

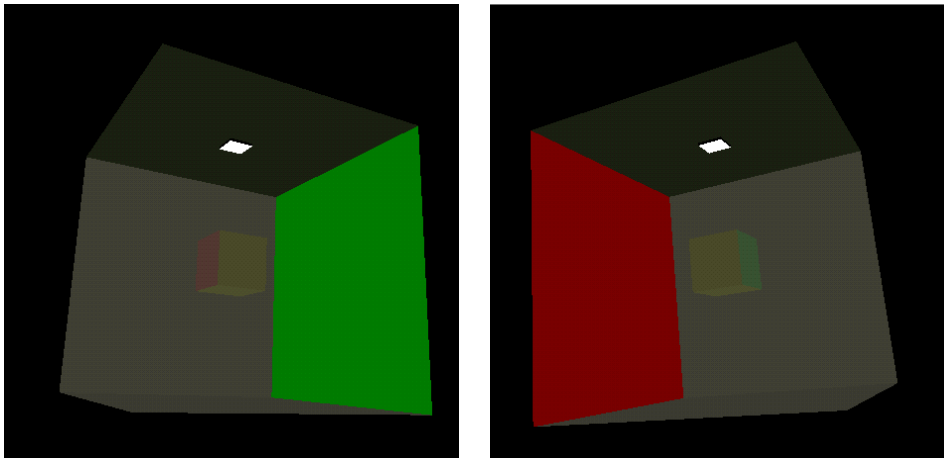


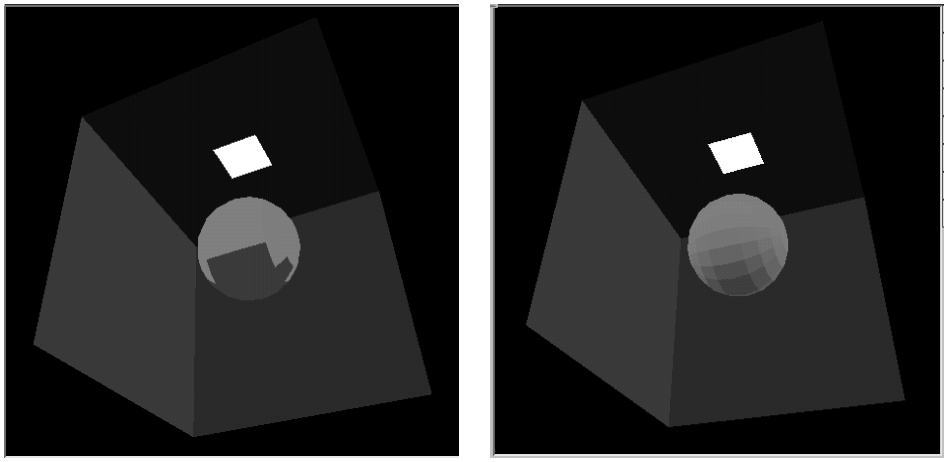
Figure 6.5: *Two views from a cubic scene with an interior white cube. The face of the cube in front of red wall looks reddish, and the face of the cube in front of green wall looks greenish. The increase in execution time due to the extended ambient term is approximately 0.05 sec.*

6.5.2 Fuzzy approach: color shifting

The fuzzy approach is specially useful to compute the illumination of curved surfaces. The scene represented in Fig. 6.6 is a clear example of the gain obtained with the fuzzy approach in front of the first approach. We have, on the left, a sharp transition between classes. On the right, using the fuzzy approach, the sharpness is avoided, obtaining a smooth transition. The reason of this is the use of the weights to express the relation between patches and classes. In Fig. 6.7 we see the same scene but with colored walls. In this case we observe, on the right image, both color bleeding and color shifting effects, not obtained with classic ambient term (left). Note that the increase in cost due to the extended ambient term is nearly negligible.

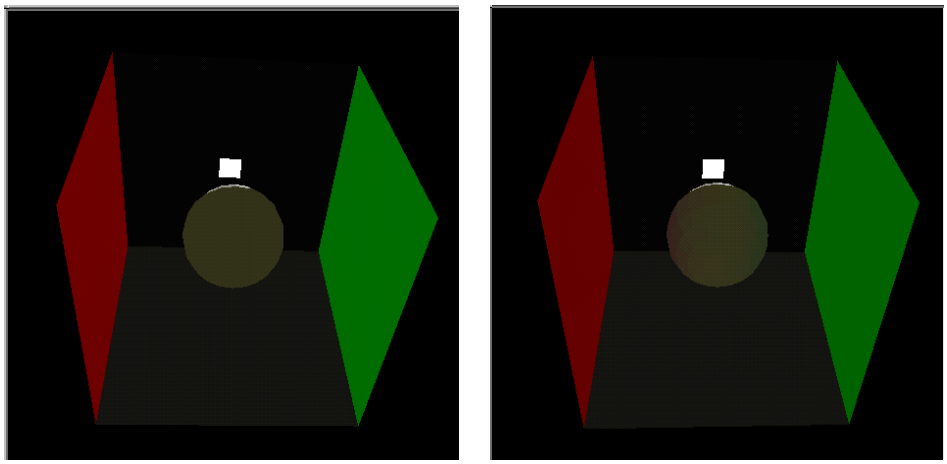
6.6 Use of a hierarchy of sub-scenes

The use of the extended ambient term in the context of a hierarchy of sub-scenes, as presented in 6.4.4, provides a greater significancy to the resulting



(a) (b)

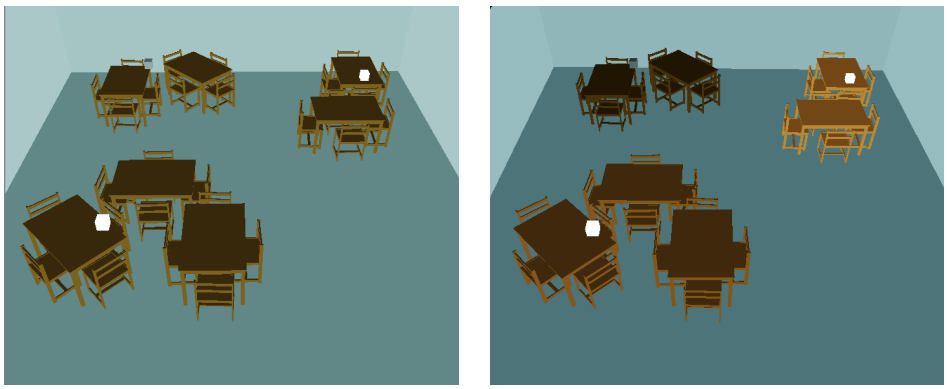
Figure 6.6: *(a) Non-fuzzy approach. Execution time: 0.23 seconds. (b) Fuzzy approach: note the smoother transition. Execution time: 0.24 seconds.*



(a) (b)

Figure 6.7: *(a) Classic ambient term. Execution time: 0.27 seconds. (b) Extended ambient term (fuzzy approach). Color bleeding and smooth transition are notable. Execution time: 0.31 seconds.*

images, since it allows to deal with the contribution of the ambient in each sub-scene. This effect can be observed in Fig. 6.8, where 3 sub-scenes including different power light sources are considered. Note that in the image obtained considering the hierarchy of sub-scenes (right) we can distinguish different levels of illumination for the sub-scenes, not observed when only considering ambient term at the level of the whole scene (left). No primary power has been expanded in this test, and no significant increase in cost has occurred when using the hierarchy.



(a) (b)

Figure 6.8: Extended Ambient Term. (a) Without using hierarchy of sub-scenes. (b) Using hierarchy of sub-scenes.

6.7 Conclusions

We have introduced the extended ambient term in this chapter. This supposes an improvement on the classic idea of ambient term. It is a new and simple method to view the undistributed (or residual) power in the solution. The main point of this new technique consists in the replacement of the constant operator that involves the classic ambient term by a more complex operator that takes into account geometric considerations. These considerations lead us to the use of a small number of classes in which the polygons in the scene are classified in a non-disjointed way according to their normal orientation, being assigned a weight for each class. A radiosity system of equations solves for the balance of power between the classes. Note that occlusion conditions are not considered, just self-emission, reflectances and normal vectors of the surfaces.

Using six classes seems suitable because it makes the classification of the polygons easier while reducing additional storage to a minimum. Although a higher number of classes could be used, the increase in cost makes this unattractive. The extended ambient term produces some nice effects that significantly improve the visual quality of the images by simulating many of the large-scale effects of a much more computationally expensive radiosity solution.

The extended ambient term provides additional gains when used together with a hierarchy of sub-scenes bounded by boxes. It makes it possible to consider a specific for each sub-scene ambient radiosity, obtaining a richer final image.

The extended ambient term has however some limitations. Large environments divided into several rooms could produce undesired effects. For instance, if we consider a scene with some light rooms and some dark rooms, the results will not be acceptable, because the ambient term will be added to both dark and light rooms. Essentially, this drawback is due to the fact that occlusion conditions are not taken into account by the extended ambient term. Note however that the same drawback exists for the classic ambient term. Note also that this drawback could be avoided by using an appropriate subdivision of the scene in a hierarchy of sub-scenes.

Because the extended ambient term is an improvement on the classic ambient term, it has the same applications. It can be applied to radiosity methods that compute the expansion solution for the rendering equation in an explicit way as in progressive radiosity [12], and in an implicit way as in the Multipath method [51]. The improvements in the obtained images can have repercussions in several fields such as design, animation production, or interior decoration. Note that the extended ambient term can be used both to rapidly obtain an image without any initial distribution of power, and as a final stage of the global illumination process to smooth the result by distributing the undistributed power.

Finally we have to remark that, although in this chapter we have only taken care of radiosity, the idea of the extended ambient term could be used in global illumination methods that deal with non-diffuse environments, like ray-tracing, z-buffer, etc. The execution for non-diffuse cases would use the average albedo instead of the reflectance ([36]) and patches would be replaced with pixels. Note that non-diffuse surfaces would show color change (also for planar polygons) depending on the view-incident direction, according to the albedo function. So, it is important to emphasize the generality of this extended ambient term in global illumination. Practically, it can complete every rendering software (z-buffer, classic ray-tracing, radiosity, etc.) adding all the nice features previously mentioned, at negligible computational cost.

Chapter 7

Conclusions and Future Research

7.1 Conclusions

We have introduced in this thesis some efficient techniques in the context of Monte Carlo radiosity. All these techniques contribute either to the reduction in computational cost (basically in the cost of casting random lines) or to the improvement in the appearance of the final image (obtained with negligible increase in computational cost).

Chapter 3 has introduced the use of quasi-Monte Carlo sequences in the context of the Multipath Monte Carlo radiosity algorithm. The main objective of this chapter is to study the performance of quasi-Monte Carlo sequences in front of classical Monte Carlo pseudo-random numbers when used in the generation of the lines needed in the Multipath algorithm. Some of the quasi-Monte Carlo sequences we have studied, like Halton, Sobol and Weyl, result in a better performance compared to classic Monte Carlo generation. These sequences have produced, in our tests, a speed-up factor close to 2, and also have presented a better asymtotical behavior. Also, we have dealt with the generation of the global uniform density of lines used in the Multipath algorithm when using quasi-Monte Carlo sequences. Finally we have studied the incidence of quasi-Monte Carlo sequences in the simulation of direct illumination, concluding, as expected, that the gain due to the use of quasi-Monte Carlo is clearly more noticeable in the distribution of direct illumination than in higher order reflections.

Chapter 4 has presented a hierarchical approach to the Monte Carlo global method for form factor computation. This approach groups objects in subscenes, resulting in a hierarchy of sub-scenes bounded by spheres. This is used to generate densities of lines adapted to each sub-scene, allowing a better use of the lines. The increase in cost due to the specificity of the lines is small in relation with the dramatic reduction of the error, resulting in clearly a better performance of the new approach (speed-up factor near to 3). Quasi-Monte Carlo sequences, added in chapter 3 to the Multipath algorithm, have also been incorporated here, resulting in a noticeable additional gain.

Chapter 5 has introduced the idea of subdividing the environment in a hi-

erarchy of sub-scenes and generating specific densities of lines to the Multipath algorithm. This results in a new algorithm, the Hierarchical Transmittance-based Multipath, that uses a bounding boxes hierarchy, establishing sub-scenes in which Multipath is locally executed. Virtual boxes bounding the sub-scenes are subdivided in virtual patches and angular regions, that act as accumulators of incoming and outgoing power, linking the different levels. The sub-scenes allow to adapt the density of lines, resulting in a reduction of the total number of global lines needed. Transmittances are calculated for each angular region. Transmittances give an idea about the transparency of boxes in each direction. They allow to accelerate global lines without loss of accuracy. Moreover, information about polygons associated to each region allows to accelerate the first shot. Finally, saving in memory the intersections lists allows to reuse the lines, thus saving in cost. This results in a speed-up factor of approximately 2 respect to classic Multipath. Moreover, we have added quasi-Monte Carlo sequences, previously studied in chapter 3, and this has reported an additional gain, reaching a speed-up factor of about 3.5 when using together both quasi-Monte Carlo and a hierarchy of sub-scenes.

The extended ambient term has been presented in chapter 6. It supposes an improvement on the classic idea of ambient term. This new technique is based on the replacement of the constant operator that involves the classic ambient term by a more complex operator that takes into account geometric considerations. We use a small number of classes (6) in which the polygons in the scene are classified in a non-disjointed way according to their normal orientation, being assigned a weight for each class. A radiosity system of equations solves the balance of power between the classes. Note that occlusion conditions are not considered, just self-emission, reflectances and normal vectors of the surfaces, so that the increase in cost is nearly negligible. The extended ambient term produces some nice effects that significantly improve the visual quality of the images. Moreover, it provides additional gains when used together with a hierarchy of sub-scenes bounded by boxes. It makes it possible to consider a specific for each sub-scene ambient radiosity, obtaining a richer final image.

Summarizing, this thesis has introduced four contributions to the Monte Carlo radiosity context. Two of them are hierarchical improvements to the global Monte Carlo method for form factors computation and the Multipath method, respectively. Another contribution involves random number generation, incorporating quasi-Monte Carlo to the global line method. The last contribution, the extended ambient term, introduces interesting nice effects in the final image with a nearly negligible cost.

7.2 Future Research

7.2.1 Parallelization of the hierarchical approaches

The main line of future research from this thesis is the parallelization of the hierarchical algorithms presented in chapters 4 and 5. This parallelization could be very advantageous when dealing with very complex scenes which involve several hierarchies of several levels each. Different sub-scenes could be distributed to different processors, making special attention to the synchronism of the processes.

7.2.2 Information theory and the hierarchical approaches

Several heuristics have been used and several empirical decisions have been taken in the hierarchical algorithms presented in chapters 4 and 5. They involve questions like the number of lines to cast in each sub-scene of the hierarchy, or, in case of the algorithm presented in chapter 5, the level of subdivision of virtual walls into virtual patches and angular regions. These questions could be solved taking into account information theory measures about the complexity of the sub-scenes [17, 18].

7.2.3 Extended ambient term in non-diffuse environments

Extended ambient term, presented in chapter 6, has been considered in the context of radiosity. However, the idea of the extended ambient is more general, and it can be used in global illumination methods that deal with non-diffuse environments, like ray-tracing, z-buffer, etc. The execution for non-diffuse cases would use the albedo function and the mean albedo instead of the reflectance ([36]) and patches would be replaced with pixels. Note that non-diffuse surfaces would show color change (also for planar polygons) according to the incident angle of view direction.

Bibliography

[1] B. Arnaldi, X. Pueyo, and J. Vilaplana. On the division of environments by virtual walls for radiosity computation. *Advances on Rendering*, pages 198–206, 1994.

[2] P. Bekaert. Hierarchical and stochastic algorithms for radiosity. *Ph.D. thesis. Katholic Univ. of Leuven*, 1999.

[3] P. Bekaert, R. Cools, and Y. Willems. An empirical comparison of monte carlo radiosity algorithms. *Proceedings of WSCG'99, vol I*, pages 9–16, 1999.

[4] P. Bekaert, L. Neumann, A. Neumann, M. Sbert, and Y. Willems. Hierarchical monte carlo radiosity. *Proceedings of the 9th Eurographics Workshop on Rendering*, pages 259–268, 1998.

[5] E. Braaten and G. Weller. An improved low-discrepancy sequence for multidimensional quasi-monte carlo integration. *J.Comp.Phisics no.33*, pages 249–258, 1979.

[6] F. Castro, R. Martinez, and M. Sbert. Quasi monte carlo and extended first shot improvements to the multipath method. *Proceedings SCCG'98, (Budmerice, Slovakia)*, 1998.

[7] F. Castro, L. Neumann, and M. Sbert. Extended ambient term. *Journal of Graphic Tools*, 5(4):1–7, 2000.

[8] F. Castro and X. Pueyo. Hierarchical improvement to the global monte carlo method for form-factors computation. *Proceedings of Computer Graphics Spanish Conference 1996*, 1996.

[9] F. Castro and M. Sbert. Aplication of quasi-monte carlo sampling to the multi-path method for radiosity. *Springer series Lecture Notes in Computational Science and Engineering, Springer*, 1999.

[10] F. Castro and M. Sbert. Quasi-monte carlo techniques in multipath radiosity. *Homage to Lluis Santaló, Càtedra Lluis Santaló d'Aplicacions a la Matemàtica*, 2002.

[11] F. Castro, M. Sbert, and L. Neumann. Hierarchical multipath. *Presented in 3rd IMACS Seminar on Monte Carlo Methods, Salzburg 2001. Submitted to Computer Graphics Forum*, 2001.

[12] M. Cohen, S. E.C., W. J.R., and D. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics, 22(4)*, pages 75–84, 1988.

[13] M. Cohen and D. Greenberg. The hemi-cube: A radiosity solution for complex environments. *Computer Graphics, 19(3)*, pages 31–40, 1985.

[14] M. Cohen and J. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, 1993.

[15] H. Faure. Good permutations for extreme discrepancy. *J.Number Theory, n.42*, pages 47–56, 1992.

[16] M. Feda and W. Purgathofer. Progressive ray refinement for monte carlo radiosity. *Proceedings of Eurographics Workshop on Rendering 1993*, pages 15–25, 1993.

[17] M. Feixas, E. Acebo, P. Bekaert, and M. Sbert. An information theory framework for the analysis of scene complexity. *Computer Graphics Forum (proc. Eurographics'99, Milan, Italy)*, 18, N.3:95–106, 1999.

[18] M. Feixas, E. Acebo, P. Bekaert, and M. Sbert. Information theory tools for scene discretisation. *Rendering Techniques'99*, pages 103–114, 1999.

[19] T. Funkhouser. Coarse-grained parallelism for hierarchical radiosity using group iterative methods. *SIGGRAPH 96 Conference Proceedings, Annual Conference Series*, pages 343–352, 1996.

[20] A. Glassner. *Principles of Digital Image Synthesis, Vol.2*. Morgan Kaufmann Pubilishers, Inc., 1995.

[21] J. Goldsmith and J. Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, 1987.

[22] C. Goral, K. Torrance, D. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. *Computer Graphics (ACM SIGGRAP Conf.Proc.)*, 18, N.3:213–222, 1984.

[23] J. Halton and G. Weller. Radical-inverse quasi-random point sequence. *Comm. ACM 7 (1964), n. 12*, pages 261–269, 1964.

[24] J. Hammersley and D. Handscomb. *Monte Carlo Methods*. Methuen and Co. Ltd., 1975.

[25] E. Hlawka. Discrepancy and riemann integration. *Studies in Pure Mathematics*, pages 121–129, 1971.

[26] D. Immel, M. Cohen, and D. Greenberg. A radiosity method for non-diffuse environments. *Computer Graphics, 20(4)*, pages 133–142, 1986.

[27] J. Kajiya. The rendering equation. *Computer Graphics Proceedings, Siggraph'86*, pages 143–150, 1986.

[28] M. Kalos and P. Withlock. *Monte Carlo Methods, Volume I*. John Wiley and Sons, 1984.

[29] A. Keller. A quasi-monte carlo algorithm for the global illumination problem in the radiosity setting. *Technical Report 260/94. University of Kaiserlautern*, 1994.

[30] A. Keller. The fast calculation of form factors using low discrepancy sequences. *Proc. of SCCG96 Budmerice*, 1996.

[31] A. Keller. Quasi-monte carlo radiosity. *Proceedings of Eurographics Workshop on Rendering*, pages 102–111, 1996.

[32] A. Keller. Quasi-monte carlo methods for photorealistic image synthesis. *Ph.D. thesis. University of Kaiserlautern*, 1997.

[33] D. Knuth. *The Art of Computer Programming, Vol.2 (Seminumerical Algorithms)*. Addison Wesley, 1969.

[34] S. Mudur and S. Pattanaik. Monte carlo methods for computer graphics. *State of the Art Report (STAR) series*, 1993.

[35] L. Neumann. Monte carlo radiosity. *Computing, 55*, pages 23–42, 1995.

[36] L. Neumann and A. Neumann. Radiosity and hybrid methods. *ACM Transactions on Graphics, vol.14(3)*, pages 233–265, 1995.

[37] L. Neumann, A. Neumann, and P. Bekaert. Radiosity with well distributed ray sets. *Proceedings of Eurographics 97*, pages 261–269, 1997.

[38] L. Neumann, W. Purgathofer, R. Tobler, A. Neumann, P. Elias, M. Feda, and X. Pueyo. The stochastic ray method for radiosity. *Rendering techniques'95*, pages 206–218, 1995.

[39] L. Neumann, R. Tobler, and P. Elias. The constant radiosity step. *Rendering Technics*, pages 336–344, 1995.

[40] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods, NSF-CBMS*. Capital City Press, 1992.

[41] S. Pattanaik and S. Mudur. Computation of global illumination by monte carlo simulation of the particle model of light. *Proceedings of third Eurographics WorkShop on Rendering*, pages 71–83, 1992.

[42] M. Pellegrini. Monte carlo approximation of form factors with error bounded a priori. *Eleventh ACM Symposium on Computational Geometry*, 1995.

[43] B. Phong. Illumination for computer generated pictures. *Communications of the ACM, 18(6)*, pages 311–317, 1975.

[44] W. Press. Numerical recipes in c. *Cambridge University Press*, 1994.

[45] R. Y. Rubinstein. Simulation and the monte carlo method. 1981.

[46] L. Santaló. *Integral Geometry and Geometric Probability*. Ed. Addison-Wesley, New York, 1976.

[47] M. Sbert. An integral geometry based method for fast form-factor computation. *Computer Graphics Forum (proc. Eurographics'93)*, 12, N.3:409–420, 1993.

[48] M. Sbert. The use of global random directions to compute radiosity. global monte carlo methods. *Ph.D. thesis. Universitat Politècnica de Catalunya, Barcelona*, 1997.

[49] M. Sbert, F. Pérez, and X. Pueyo. Global monte carlo. a progressive solution. *Rendering Techniques '95. Springer Wien New York*, pages 231–239, 1995.

[50] M. Sbert and X. Pueyo. Integral geometry methods for form-factor computation. *Proc. of the VI Encuentros de Geometría Computacional*, pages 297–305, 1995.

[51] M. Sbert, X. Pueyo, L. Neumann, and W. Purgathofer. Global multi-path monte carlo algorithms for radiosity. *The Visual Computer*, pages 47–61, 1996.

[52] P. Shirley. Time complexity of monte carlo radiosity. *Proceedings of Eurographics 91*, pages 459–465, 1991.

[53] Y. Shreider. *The Monte Carlo Method*. Pergamon Press, 1966.

[54] F. Sillion. Radiosity with non-diffuse reflectors. *ACM SIGGRAPH'93 Course Notes- Making Radiosity Practical*, pages chapter 5, 1–25, 1993.

[55] F. Sillion and C. Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Pubilishers, Inc., 1994.

[56] C. Soler. Hierarchical instantiation for radiosity. *Proceedings of Eurographics Rendering Workshop*, pages 173–184, 2000.

[57] C. Soler and F. Sillion. Accurate error bounds for multi-resolution visibility. *Proceedings of Workshop on Rendering*, 1996.

[58] H. Solomon. Geometric probability, siaam-cbms 28. 1978.

[59] J. Spanier. Quasi-monte carlo methods for particle transport problems. *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 121–148, 1995.

[60] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. Springer, Berlin, 1980.

[61] J. Struckmeier. Fast generation of low-discrepance sequences. *J.Comp.Appl.*, pages 29–41, 1995.

[62] L. Szirmay-Kalos, B. Csebfalvi, and W. Purgathofer. Importance driven quasi-random walk solution to the rendering equation. *Proceedings of WSCG 98*, pages 378–385, 1998.

[63] L. Szirmay-Kalos, T. Foris, L. Neumann, and B. Csebfalvi. An analysis of quasi-monte carlo integration applied to the transillumination radiosity method. *Proceedings of Eurographics 97*, 1997.

[64] L. Szirmay-Kalos and W. Purgathofer. Analysis of the quasi-monte carlo integration of the rendering equation. *Technical report TR-186-2-98-22*, 1999.

[65] L. Szirmay-Kalos, M. Sbert, R. Martinez, and R. Tobler. Incoming first-shot for non-diffuse global illumination. *Proceedings of SCCG*, 2000.

[66] J. Wallace, K. Elmquist, and E. Haines. A ray tracing algorithm for progressive radiosity. *SIGGRAPH 89 Conference Proceedings, Annual Conference Series*, pages 315–324, 1989.